LOAN DOCUMENT

PHOTOGRAPH THIS SHEET

DTIC ACCESSION NUMBER

LEVEL

INVENTORY

DOCUMENT IDENTIFICATION

H A N D L E W I T H C A R E

DISTRIBUTION STATEMENT

ACCESSION FOR
NTIS GRA&I ☒
DTIC TRAC ☐
UNANNOUNCED ☐
JUSTIFICATION

BY
DISTRIBUTION/
AVAILABILITY CODES

DISTRIBUTION	AVAILABILITY AND/OR SPECIAL	
A-1		

DISTRIBUTION STAMP

DATE ACCESSIONED

Reproduced From
Best Available Copy

DATE RETURNED

19981223 014

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NUMBER

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC

DTIC FORM JUN 90 70A

DOCUMENT PROCESSING SHEET

PREVIOUS EDITIONS MAY BE USED UNTIL STOCK IS EXHAUSTED.

LOAN DOCUMENT

HC 12-18-78

UNCLASSIFIED

NO DISTRIBUTION STATEMENT

NADC Tech. Info.

APPENDIX 11

ANALYSIS RETURN, CP PROCESSOR

FINAL SOFTWARE REPORT

DATA ITEM NO. A005

INTEGRATED ELECTRONIC WARFARE SYSTEM ADVANCED DEVELOPMENT MODEL (ADM)

7800987-12

PREPARED FOR:

NAVAL AIR DEVELOPMENT CENTER
WARMINSTER, PENNSYLVANIA

CONTRACT N62269-75-C-0070

RAYTHEON
ELECTROMAGNETIC SYSTEMS DIVISION

1 OCTOBER 1977

UNCLASSIFIED

APPENDIX 11

CLASSIFICATION PROCESSOR, ANALYSIS RETURN, DESIGN SPECIFICATION

FINAL SOFTWARE REPORT

DATA ITEM A005

INTEGRATED ELECTRONIC WARFARE SYSTEM (IEWS)
ADVANCED DEVELOPMENT MODEL (ADM)

Contract No. N62269-75-C-0070

Prepared for:

Naval Air Development Center
Warminister, Pennsylvania

Prepared by:

RAYTHEON COMPANY
Electromagnetic Systems Division
6380 Hollister Avenue
Goleta, California 93017

1 OCTOBER 1977

RAYTHEON	RAYTHEON COMPANY LEXINGTON, MASS. 02173	CODE IDENT NO. 49956	SPEC NO. 53959-GT-0761 SHEET 1 OF 103 REV 103

TYPE OF SPEC

TITLE OF SPEC

COMPUTER SUBPROGRAM DESIGN DOCUMENT,
ANALYSIS RETURN PROCESSING, IEWS, ADM

FUNCTION	APPROVED	DATE	FUNCTION	APPROVED	DATE
WRITER-	Thomas L. Chiarello	17 Nov 76			
	George D. Alimonti Jr.	12/7/76			

REVISIONS

CHK	DESCRIPTION	REV	CHK	DESCRIPTION	REV

REVISION																						
SHEET NO.																						
REV STATUS OF SHEETS	REVISION																					
	SHEET NO.																					

10-1348 (11/68) VELLUM PRINTED IN U.S.A.
10-1348 (11/68) FILM

TABLE OF CONTENTS

1.0 SCOPE

1.1 IDENTIFICATION

This document describes the implementation of the Analysis Return Functional Group (ARFG) of the SC Operational Software resident in the Classification Processor (CP).

1.2 SUBPROGRAM TASKS

1.2.1 Analysis Return Driver (ANDR)

ANDR shall have the responsibility of decoding Analysis Return messages. These messages originate in the Analysis Processor and are the response to an analysis request. After decoding, ANDR shall call the appropriate Analysis Return processing routine.

1.2.2 New Emitter Processing 2 (ANNE2)

ANNE2 shall process Analysis Return messages which have a return module code of 1. These messages are the result of deinterleaving analysis requests from New Emitter Processing 1 (see Sorter Message Processing Design Document 53959-GT-0755).

1.2.3 New Emitter Processing 3 (ANNE3)

ANNE3 shall process Analysis Return messages which have a return module code of 2. These messages are the result of contemporaneous analysis requests from ANNE2.

1.2.4 NOFA 2 Processing 2 (ANNA2)

ANNA2 shall process Analysis Return messages which have a return module code of 3. These messages are the results of scan analysis requests from NOFA2 Processing 1 (see Sorter Message Processing Design Document, 53959-GT-0755).

1.2.5 NOFA 2 Processing 3 (ANNA3)

ANNA3 shall process Analysis Return messages which have a return module code of 4. These messages are the result of contemporaneous analysis requests from ANNA2.

1.2.6 Emitter of Concern (EOC) Processing 2 (ANOC2)

ANOC2 shall process Analysis Return messages which have a return module code of 5. These messages are the result of scan analysis requests from EOC Processing 1 (see Sorter Message Processing Design Document, 53959-GT-0755).

1.2.7 EOC Processing 3 (ANOC3)

ANOC3 shall process Analysis Return messages which have a return module code of 6. These messages are the result of contemporaneous analysis requests from ANOC2.

1.2.8 EOC Processing 4 (ANOC4)

ANOC4 shall process Analysis Return messages which have a return module code of 7. These messages are also the result of contemporaneous analysis requests from ANOC2.

1.2.9 Emitter Classification 2 (ANEC2)

ANEC2 shall process Analysis Return messages which have a return module code of 8. ANEC2 shall be the principal routine for accomplishing the second level of emitter classification, namely, the elimination of candidates on the basis of scan type and scan period from a list created by Emitter Classification 1 (see Emitter Classification 1 Design Document, 53959-GT-0760).

1.2.10 Emitter Classification 3 (ANEC3)

ANEC3 shall process Analysis Return messages which have a return module code of 9. These messages are the result of contemporaneous analysis requests from ANEC2.

10-134B CONT (10/68) VELLUM PRINTED IN U.S.A.
10-134B CONT (10/68) FILM

2.0 APPLICABLE DOCUMENTS

The following documents, of the exact issue shown, form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of the Computer Program Design Specification for the Integrated Electronic Warfare System (IEWS) Advanced Development Model (ADM) Program shall be considered superseding requirements.

2.1 COMPUTER PROGRAM PERFORMANCE SPECIFICATION

Computer Program Performance Specification for the Integrated Electronic Warfare System (IEWS) Advanced Development Model (ADM) Program (U), Raytheon Company, Electromagnetic Systems Division, (Number 061290529), (date 1 June 1976), (classification U).

2.1.1 Applicable CPPS Paragraphs

Analysis Return Driver	Not Specified Explicitly
New Emitter Processing 2	3.3.2.1.2.1
New Emitter Processing 3	3.3.2.1.2.1
NOFA 2 Processing 2	3.3.2.1.2.2
NOFA 2 Processing 3	3.3.2.1.2.2
EOC Processing 2	3.3.2.1.2.2
EOC Processing 3	3.3.2.1.2.2
EOC Processing 4	3.3.2.1.2.2
Emitter Classification 2	3.3.3
Emitter Classification 3	3.3.3

2.2 COMPUTER PROGRAM DESIGN SPECIFICATION

Computer Program Design Specification for the Integrated Electronic Warfare System (IEWS) Advanced Development Model (ADM) Program (U), Raytheon Company, Electromagnetic Systems Division, (Number 53959-GT-0750), (2 September 1976), (classification U).

2.3 DATA BASE DESIGN DOCUMENT

The Common Data Base Design Document, System Controller Unit, IEWS, ADM, document No. 53959-GT-0751, shall apply to this subprogram.

2.4 MISCELLANEOUS DOCUMENTS

The following documents shall apply to this subprogram:

Document No.	Document Title
53959-GT-0756	Computer Subprogram Design Document, Executive, IEWS, ADM
53959-GT-0755	Computer Subprogram Design Document, Sorter Message Processing, IEWS, ADM
53959-GT-0760	Computer Subprogram Design Document, Emitter Classification 1 IEWS, ADM
53959-GT-0759	Computer Subprogram Design Document, Data Extraction, IEWS, ADM
WS-8506 Revision 1, 1 November 1971	Requirements for Digital Computer Program Documentation

3.0 REQUIREMENTS

3.1 SUBPROGRAM DETAILED DESCRIPTION

3.1.1 Analysis Return Driver (ANDR)

ANDR shall be the driver routine of the Analysis Return Functional Group. The EXEC shall pass to ANDR a pointer to the Analysis Return message ANMNO word (see Figure 1), in the X-register. ANDR shall use the pointer to retrieve the return module code from the message. The code shall be verified to be a valid code and then used as an index to the Analysis Return Processing table (ANMPT). The index (equal to the value of ANRMC) shall be added to the base address of the table and this address shall be used indirectly to call one of the Analysis Return processing routines (whose list of symbolic names constitute ANMPT). Each of the Analysis Return processing routines shall receive the same input:

1) The address of the Analysis Return message word 2 in the X-register.
2) The emitter file number from the message in the A-register.

After the Analysis Return processing routine has completed its task, control shall be returned to the driver. If the Analysis Return processing routine has returned via return 1, the X-register shall contain a pointer to an analysis request message buffer and the Executive message function shall be called to output the analysis request (label AND9Ø). Control shall then be returned to the EXEC. If the Analysis Return processing routine has returned via return 2, control shall be returned to the EXEC.

3.1.2 New Emitter Processing 2 (ANNE2)

ANNE2 shall perform the following tasks:

(a) Calculate the address (EFP) of the ETF entry to be processed.

(b) Accept and process the results of deinterleaving in PRI Test 2 (ANPT2).

(c) Assess PW quality and check for presence of long pulse data.

(d) Perform a check for emitters having harmonically related PRI's in Harmonic PRI Test 1 (ANHP1).

(e) Pass an analysis request message for contemporaneous analysis to the analysis return driver (ANDR).

To accomplish this, ANDR shall call ANNE2 with a pointer to word 2 of the Analysis Return message stored in the X-register. The A-register shall contain the emitter file number (EFN) in the least significant byte.

ANNE2 shall immediately call subroutine SOGET which shall compute the address of the emitter track file (ETF) entry and shall return it in the B-register as EFP. ANNE2 shall then call subroutine PRI Test 2 (ANPT2) to process the results of deinterleaving.

The subroutine PW Test (ANPWT) shall be called by ANNE2 to assess the validity of the PW data and to check for long pulse data. ANNE2 shall test the return from ANPWT to determine if long pulse data has been detected. If long pulse data is indicated, the PW validity (EFPWV) bit in the ETF shall be reset to zero and then ANNE2 shall proceed. (At this point a time-out period shall be initiated and a return made when long pulse processing is implemented). If long pulse data is not indicated, the

processing shall continue directly to call Harmonic PRI Test 1 (ANHP1). (ANHP1 will check for the presence of emitters with harmonically related PRI's when contemporaneous analysis is implemented).

ANNE2 shall test the return from ANHP1 to determine if contemporaneous analysis (CA) should be requested. If CA is not required, ANNE2 shall reset the analysis wanted (AW) bit and set the CA Request (CAR) bit in the analysis request message. If CA is required, ANNE2 shall set both the AW and CAR bits. (CA required path shall never be executed until CA is implemented). In either case, ANNE2 shall store the EFN in the analysis request message. ANNE2 shall load the address of the analysis request message into the X-register and shall return to the analysis return driver (ANDR).

3.1.2.1 PRI Test 2

The logic flow for PRI Test 2 (ANPT2) shall be as shown in 3.2.2.1. ANPT2 shall be a direct return to the calling routine. (This is a dummy subroutine which will be enhanced when deinterleaving is implemented).

3.1.2.2 Pulse Width Test

The logic flow for PW Test (ANPWT) shall be as shown in 3.2.2.2. A subroutine call shall be made to ANPWT with the address of the ETF entry, EFP, in the B-register. ANPWT shall extract the PW code, EFPW (EFP), from the ETF and shall test for the value B'1111'. If EFPW (EFP) = B'1111', then the long pulse indication EFLP (EFP) shall be set to 1 and a normal return shall be made to the calling routine indicating the presence of long pulse data. If EFPW (EFP) ≠ B'1111', the PW quality factor, EFQPW (EFP) shall be tested. If EFQPW (EFP) = B'1111', indicating bad quality, the PW validity bit, EFPWV (EFP), shall be reset to zero. If EFQPW (EFP) ≠ B'1111', then EFPWV (EFP) shall be set to 1. For either result

of the EFQPW test, the return address shall be incremented by one to indicate that no long pulse data is present. ANPWT shall then return to the calling routine.

3.1.2.3 Harmonic PRI Test 1

The logic flow for Harmonic PRI Test 1 (ANHP1) shall be as shown in 3.2.2.3. ANHP1 shall increment the return address by one to indicate no contemporaneous analysis required and shall return. (When contemporaneous analysis is implemented, ANHP1 shall be enhanced).

3.1.3 New Emitter Processing 3 (ANNE3)

ANNE3 shall perform the following tasks:

(a) Calculate the address (EFP) of the ETF entry to be processed.

(b) Accept and process the results of contemporaneous analysis (CA) in Harmonic PRI Test 2 (ANHP2).

(c) Assess frequency quality and store result in ETF by calling Frequency Test (ANFQT).

(d) Output a classification message to the Executive and return to the Analysis Return Driver (ANDR).

To accomplish this, ANDR shall call ANNE3 with a pointer to word 2 of the Analysis Return message in the X-register. The A-register shall contain the emitter file number (EFN) in the least significant byte.

ANNE3 shall immediately call subroutine SOGET which shall compute the address of the emitter track file (ETF) entry and shall return it in the B-register as EFP. ANNE3 shall then call Harmonic PRI Test 2 to process the results of contemporaneous analysis (CA).

The subroutine Frequency Test (ANFQT) shall be called by ANNE3 to assess the validity of the frequency data. Upon return from ANFQT, ANNE3 shall output a classification message to the Executive. The X-register shall contain a pointer to the first word in the classification message. ANNE3 shall then return to the analysis return driver (ANDR).

3.1.3.1 Harmonic PRI Test 2

The logic flow for Harmonic PRI Test 2 (ANHP2) shall be as shown in 3.2.3.1. ANHP2 shall be a direct return to the calling routine. (This is a dummy subroutine which will be enhanced when CA is implemented).

3.1.3.2 Frequency Test

The logic flow for the Frequency Test (ANFQT) shall be as shown in 3.2.3.2. A subroutine call shall be made to ANFQT with the address of the ETF entry (EFP) in the B-register. ANFQT shall establish a local data area to store PARAM, M, and QVAL in consecutive locations. ANFQT shall set PARAM equal to the value of EFFREQ (EFP). PARAM shall be tested for the presence of all 1's which shall be the default frequency value if no IFMR output occurs. If the default value is detected, the frequency validity, EFV (EFP), shall be reset to zero to indicate bad frequency data and ANFQT shall return. If the default value is not detected, processing shall proceed by setting M to 15 and QVAL to the value of EFQF (EFP).

ANFQT shall call parameter quality test (SOQUT) with a pointer in the X-register to PARAM. SOQUT shall return with an indication of good quality (GDQ) contained in the A-register. The value of GDQ shall be stored in the frequency validity bit EFV (EFP). ANFQT shall return to the calling routine.

10-1349 CONT (11-68) VELLUM PRINTED IN U.S.A.
10-1349 CONT (11-68) FILM

3.1.4 NOFA2 Process 2 (ANNA2)

3.1.4.1 ANNA2

ANNA2 shall be called by the Analysis Return Driver (ANDR), if the return module code of the analysis return data (AR data) is X'03' (See Figure 1). The driver shall pass to ANNA2 the address of AR data word 2 in the X-register and the emitter file number (EFN) in the A-register. SOGET (see Sorter Message Processing CSDD) shall immediately be called to convert EFN to the address of an Emitter Track File entry (EF entry). The address shall be returned by SOGET in the B-register. The scan type, as determined by the scan analysis module, shall be retrieved from the AR data. If the scan type is "sidelobe", processing shall continue at label ANN1Ø. If the scan type indicates a null measurement, the return address on the stack shall be incremented by 1 so that subroutine return No. 2 is performed (label ANN9Ø). Control shall then be returned to the AR driver.

3.1.4.2 Subroutine Returns from ANNA2

Two returns from ANNA2 shall be possible:

1) AR driver shall output an analysis request message to the EXEC. A pointer to the message buffer shall be returned to the AR driver in the X-register.
2) AR driver shall not output any Analysis Request message to the EXEC.

3.1.4.3 ANN1Ø

The scan state indicator (EFSIND) shall be retrieved from the EF entry. If this indicator is Ø, it shall be set equal to 1 (in the EF entry) and control shall be returned to the AR driver (via ANN9Ø). If EFSIND is not Ø, processing shall continue at label ANN2Ø.

3.1.4.4 ANN2Ø

The scan type (EFSTYP) shall be retrieved from the EF entry and from the analysis return data (ANSTY). The two codes shall be compared. If they are not equal, processing shall continue at label ANN3Ø. If they are equal, the scan period (EFSPRD) shall be retrieved from the EF entry and from the analysis data (ANSPR). The absolute value of the difference of the two scan periods shall be computed. If this difference is less than, or equal to, the constant Δ_{SPRD}, control shall be returned to the AR driver (via ANN9Ø). Otherwise, processing shall continue at label ANN3Ø.

3.1.4.5 ANN3Ø

The value of scan type code (EFSTYP) in the EF entry shall be set to that of the AR data (ANSTY). Similarly, for the scan period (EFSPRD). An attempt shall then be made to reclassify this "changed" emitter. To do this, the emitter file number (ANEFN) shall be retrieved from the AR data and saved in the Analysis Request message buffer (ANNCA). ANEFN shall then be passed to the Level 1 search module (ECLV1) in the X-register (see Emitter Classification 1 CSDD). If this routine finds no candidates (1st return from ECLV1), control shall be returned to the AR driver via ANN9Ø. If there are candidates (2nd return from ECLV1), a pointer to the candidate list (in the common data base) shall be returned in the X-register. The Level 2 Search Module (ANLV2) shall receive this pointer to the candidate list as input and shall output a refined candidate list. If there are no candidates as a result of the Level 2 Search, control shall be returned to the AR driver via ANN9Ø. If candidates still exist, the pointer to the refined candidate list shall be saved in the Analysis Request message buffer, ANNCA. (See Figure 2). The New Emitter Link Analysis 1 module (ANEL1) shall be called to determine if contemporaneous analysis is required.

ANEL1 shall be a dummy routine in the priority 1 implementation. If contemporaneous analysis is required (return 2, which is never executed in priority 1), the analysis wanted bit (ANAW) shall be set in the Analysis Request message buffer, ANNCA. If not required (return 1), the ANAW bit shall be cleared. If either case, return 1 shall be performed to return control to ANDR, with the address of ANNCA in the X-register.

3.1.5 NOFA2 Process 3 (ANNA3)

3.1.5.1 ANNA3

This routine shall be called by the Analysis Return driver (ANDR), if the return module code of the analysis return data (AR data) is X'Ø4' (See Figure 1). The driver shall pass to ANNA3 the address of the AR data in the X-register. The New Emitter Link Analysis 2 routine (ANEL2) and the Family Association routine (ANFAM) shall immediately be called. Then the Ambiguity Resolution (ANAMB) shall be called. This routine shall be passed a pointer to the AR data in the X-register. Finally, the return-to-AR-driver address (on the stack) shall be incremented, so that ANNA3 will never cause any analysis request messages to be sent to the EXEC by the AR driver.

3.1.5.2 Subroutine Returns from ANNA3

ANNA3 shall always cause the instruction after the call to ANNA3 to be skipped. The returns from ANNA3 shall be:

1) Null. Never executed.
2) AR driver shall not output any Analysis Request message to the EXEC.

3.1.6 EOC Process 2 (ANOC2)

3.1.6.1 ANOC2

This routine shall be called by the Analysis Return driver (ANDR), if the return module code of the analysis return data is X'Ø5' (See Figure 1). The driver shall pass to ANOC2 the address of the AR data in the X-register and the EFN in the A-register. The EFN shall be saved in the update message buffer, ANUPM (See Figure 4) and in the analysis request buffer, ANOCA, (See Figure 2). SOGET shall then be called to convert EFN into an EF entry address, which shall be returned in the B-register.

The Scan Test 2 (ANST2) routine shall then be called. It shall receive as input the pointer to the AR Data in the X-register. Upon return, the pointer to the candidate list (ANPTR) shall be retrieved from the AR data. This shall be passed as input to the Level 2 Search routine (ANLV2) in the X-register. If Level 2 Search finds candidates (2nd return), a pointer to the refined candidate list shall be returned in the X-register and control shall be transferred to label ANC3Ø. Otherwise, processing shall continue at label ANC1Ø.

3.1.6.2 ANC1Ø

The platform link pointer (EFPLNK) shall be retrieved from the EF entry. A test shall be made to see if EFPLNK is equal to the emitter file number. If not equal, the emitter is "platform linked" and the Delete Link Processing (SODLK) routine shall be called. Otherwise, the call to SODLK shall be skipped. The EXEC shall then be called to ouptut an update message, ANUPM (See Figure 4). Finally, the return-to-AR-driver address on the stack shall be incremented and subroutine return No. 2 (do not output any analysis request) shall be performed to return control to the AR driver.

3.1.6.3 ANC3Ø

The pointer to the refined candidate list shall be saved in the contemporaneous analysis (CA) request message buffer (ANOCA). The emitter file number (CLEFN) shall be retrieved from the candidate list and saved in the CA request buffer. The identification code (EFID) shall be retrieved from the EF entry. This code shall then be compared to the identification code of each candidate in the list. If there is no match, processing shall continue at label ANC5Ø. If there is a match, the old EF entry id code is still valid. The return module code (ANRMC) in the CA request buffer (ANOCA) shall be set to X'Ø7', to indicate EOC Process 4 as the analysis return module. The Update Link Analysis 1 routine (ANUL1) shall then be called to determine if contemporaneous analysis is required. If required (return 2), the ANAW bit in the analysis request buffer shall be set (label ANC6Ø). If analysis is not required (return 1), the ANAW bit shall be cleared (label ANC7Ø). In either case, subroutine return No. 1 (output the analysis request) shall be performed to return control back to the AR driver, with the address of the analysis request buffer in the X-register.

3.1.6.4 ANC5Ø

The return module code (ANRMC) in the CA request buffer (ANOCA) shall be set to X'Ø6', to indicate EOC Process 3 as the analysis return module. The New Emitter Link Analysis 1 routine (ANEL1) shall then be called to determine if contemporaneous analysis is required. If required (return 2), control shall be sent to label ANC6Ø (described above). If not required (return 1) label ANC7Ø (also described above).

3.1.6.5 Subroutine Returns from ANOC2

Same as 3.1.4.2

3.1.6.6 Update Link Analysis 1 (ANUL1)

ANUL1 shall be a dummy routine in the priority 1 implementation. ANUL1 shall always perform return 1 to the calling routine. This return shall indicate that no contemporaneous analysis is required.

3.1.7 EOC Process 3 (ANOC3)

3.1.7.1 ANOC3

See 3.1.5

Same as ANNA3, except for the fact that the routine shall be called by the Analysis Return driver (ANDR), if the return module code of the AR data is X'Ø6' (see Figure 1).

3.1.8 EOC Process 4 (ANOC4)

3.1.8.1 ANOC4

This routine shall be called by the Analysis Return driver (ANDR), if the return module code of the analysis return data (AR data) is X'Ø7' (See Figure 1). The driver shall pass to ANOC4 the address of the AR data in the X-register and the EFN in the A-register. The EFN shall be passed in the A-register to the Update Link Analysis 2 (ANUL2) routine. ANUL2 shall determine if there is any platform linkage change.
If there is change (return 2), processing shall continue at label ANK9Ø.
If no change is detected (return 1), the emitter file number shall be saved in the update message buffer (ANUPM). This message shall then be sent to the EXEC. Processing shall continue at ANK9Ø.

3.1.8.2 ANK9Ø

The return-to-AR-driver address on the stack shall be incremented so that ANOC4 shall always perform the "no analysis" return to the AR driver. Control shall then be returned to the AR driver.

3.1.8.3 Subroutine Returns from ANOC4

The returns from ANOC4 shall be:

1) Null. Never excuted.
2) AR driver shall not output any Analysis Request message to the EXEC.

3.1.8.4 Update Link Analysis 2 (ANUL2)

ANUL2 shall receive the EFN in the A-register. SOGET shall immediately be called to convert EFN to an EF entry address. The function of ANUL2 shall be to determine if there has been any change in the platform linkage of the emitter. This function has not been implemented. The abbreviated priority 1 implementation shall merely set the platform link in the Emitter track file entry for this emitter to the emitter file number (EFN), i.e., no platform links. Control shall then be returned to the calling program.

3.1.8.4.1 Subroutine Returns from ANUL2 - The returns from ANUL2 shall be:

1) No platform linkage changed detected.
2) Platform linkage change detected.

In the abbreviated implementation, return 1 shall always be performed.

3.1.9 ANEC2 Emitter Classification 2

This is the principal subroutine for accomplishing the second level of emitter classification, namely: eliminating candidates from the list created by ECDR and its subroutines, on the basis of scan type (exact match) and scan period (between limits match). As such, it is largely a logical skeleton, most of the aforesaid task being accomplished by its dependent subroutines (described hereafter).

Upon entry X-register points to word 1 of a 3-word block:

	15–8			7–0
X →	- dc -			EFN
	CLAD			
	ST (15–12)	dc	(9)	SPD (0)

EFN: Emitter Track File # to which the candidate list applies.

CLAD: Candidate List Address

ST, SPD: Scan Type & Scan Period as obtained from a scan analysis request executed on behalf of Emitter Track File # EFN after ECDR was called for EFN and before the present call on ANEC2 for EFN.

The steps executed are as follows. (Note: Steps are keyed to to program labels and unlabelled blocks preceded by a numbered comment line, e.g., "; 22".)

ANEC2 - Call ANST2 with X-register as described above. X is unchanged on return.

; 2 - Save A and B-registers on stack. Fetch 2nd word (CLAD) of input block store it in contemp. analysis request block (CRCLAD). Save a copy of CLAD in X-register.

; 3 - Call ANLV2. Most of the winnowing down is done here. If no candidates survive, return to call +1 (To increment return-to-driver address to call +2 and go to step "Done") else to call +2 (Next).

; 4 - X-register still contains CLAD. Fetch (CLAD) = EFN in right byte. Build byte-split word with EFN still in right byte and ANEC2's return module code (RMCEC2) in the left byte. Store result in contemp. analysis request block (CRRMCD).

; 5 - Set A-register = X'8ØØ8'. This will be the contemp. analysis request word if the forthcoming call to ANEL1 indicates that analysis will be wanted. (Bit 15 is "Analysis Wanted" bit, Bit 3 indicates analysis type is Contemp.

Set B-register = X'8ØØØ' This will be used upon analysis-not-wanted return from ANEL1 to wipe out the analysis-wanted bit in A-register.

Call ANEL1. If contemp. analysis is not wanted, return to call +1 (Step 6) else to call +2 (Step 7).

; 6 - Contemp. Analysis not wanted: Use B-register to wipe out bit 15 of A-register.

; 7 - Store request word now in A-register in the Contemp.-Analysis request block (CRREQW).

Set X-register = Address of 1st word of Contemp. Analysis request block = CRQMSG.

Done - Entered from Step 3 (No candidates left) or Step 1.

Restore B and A-registers from stack.

Return.

3.1.9.1 ANST2 - Subroutine of ANEC2 Scan Test 2

ANST2 tests the existing ETF scan type (ESTY) against certain standard types and under certain conditions alters both the ETF scan type (ESTY) and scan period (ESPD).

Upon entry X-register is exactly as described for the entry to ANEC2. The steps are as follows:

ANST2 - Save A, B and X-registers on stack.

; 1 - Fetch 1st word of input block = ((X)) with EFN in right byte. Mask out left byte and call SOGET. This is a subroutine in Sorter Message Processing (Document No. 53959-GT-0755). That computes

$$\text{B-Reg} \leftarrow \text{ETF} + 16 * \text{EFN}$$

where EFN is in A-Reg.

; 2 - Fetch word containing scan type field ST in input block ptd to by X-reg, mask off extraneous fields of word and compare ST to sidelobe scantype code (SIDLOB). If not equal, go to step LKNMC else next.

; 3 - Move Addr ETF (EFN) = ETF + 16 * EFN now in B-reg into X-reg and call ECSTC. This is a subroutine shared with ECST1 (Scan Test 1) in Emitter Classification 1 (Driver ECDR - Document No. 53959-GT-0760). Return is always to call +2.

DONE - This step is entered from Steps 3 (above), LKNMC (below), and 4 (below).

Restore X, B and A-registers.

Return.

LKNMC - (Look at Null - Measure Code)

This step is entered from Step 2 if ST $\neq$ sidelobe scan-type code.

Compare ST to Null-Measure Scan-type code. If equal, go to Step DONE, else next.

; 4 - Pick up 3rd word of input block (containing latest analysis return values for scan type & period) and use it to update the ETF word containing the same two items (ESTYD Rel to B-register).

Go to step DONE.

3.1.9.2 ANLV2 - Subroutine of ANEC2 Emitter Classification Level 2 Search

This is the work horse subroutine of ANEC2. It's chief function is to eliminate candidates from an existing candidate list on the basis of scan type and period comparisons.

It is entered with X-Reg = Address or existing candidate list. The process is carried out "in place" so that the resulting, reduced candidate list is stored at the same address (which is also exit value of X-Reg). If there are no candidates, special actions are taken; these occur in steps NOCAND through 13.

ANLV2 - Save A, B, E and X-Reg's on stack. The saved X-Reg on the stack will be referred to as clad (Lower case indicating contents of location CLAD relating to S-Reg when the stack map is computed)

;1 - A-Reg $\leftarrow$ ((X)) and X $\leftarrow$ X + 1
X-Reg which was pointing to the Cand. list header word now points to the first Cand. list entry. Push X-Reg twice to the stack for later use as rdpt (Read point) and stpt (Store point).

;2 - The Cand. list header word:

NCAND	EFN
15 8	7 0

Was loaded into A-Reg in Step 1. Now separate the bytes so that EFN ends up in A-Reg and NCAND (Right justified) ends up in E-Reg.

Push E-Reg twice to the stack for later use as ncand (Loop iteration control) and as nleft (Number of candidates not [yet] eliminated).

; 3 - Call SOGET. This is a subroutine in Sorter Message Processing (Document No. 53959-GT-0755) that computes

B-Reg ◄— ETF + 16 * EFN

where EFN is in A-Reg.

Push returned value of B-Reg to stack for (possible) later use as adef (step NOCAND).

Pick up ETF word containing both scan type (ESTY) and scan period (ESPD) - - at ESTYD relative to B-Reg.

; 4 - The word just loaded into A-Reg is:

15 – 12	12 – 9	9 – 0
ESTY		ESPD

Unpack so that ESPD is in A-Reg and ESTY (right justified) is in X-Reg (Mask: ESTYM; SHIFT: ESTYS). Push A-Reg to stack for later use as spud.

; 5 - Pick up and push to stack (ODA. ST + ESTY + 1) ≡ ((B) + (X) + 1). This will later be used as higr = the largest group # (index into EL2) that has scan type ESTY.

Pick up and push to stack (ODA. ST + ESTY) ≡ ((B) + (X)). This will later be used as logr = the smallest group # that has scan type ESTY.

Winnow - This step begins the major loop of the subroutine. It is entered once from above (Step 5) and N-1 times from step tally where N = original value stored in ncand in Step 2. All stops from the present, down to and including step tally are within the loop.

Fetch to A-Reg the next cand. list entry = (rdpt).

rdpt ◄— rdpt + 1

Make a copy of A-Reg in X-Reg for use in step keep.

Mask out the ident field in A-Reg so that A = group # of current Cand. List entry.

Compare this group # to logr (see step 5).

If group # < logr go to step CANCEL - else next

; 6 - Compare group # to higr

If group # ≥ higr go to step CANCEL - else next

; 7 - Compute in B-Reg the address of the file in EL2 whose index is group # =

EL2 + 11 * (Group # - 1)

(Done by Call E2ADR)

Double load to A and E-Reg's from MXSND relative to B-Reg. This puts the maximum scan period (MXSN) in A-Reg and the minimum scan period (MNSN) in E-Reg. Both fields occupy BITS Ø - 9 and require masking.

; 8 - Mask out extraneous fields leaving A-Reg = MXSN (Mask: ESPDM).

Compare MXSN to spud (see step 4).

If spud > MXSN go to step CANCEL - else next

; 9 - Mask out extraneous fields leaving E-Reg = MNSN.

Compare MNSN to spud (loaded into A-Reg.)

If spud ≥ MNSN go to step keep - else next

Cancel - This step is entered from steps

Winnow	-	Group #	<	logr
6	-	Group #	≥	higr
8	-	spud	>	MXSN
9	-	spud	<	MNSN

Do: nleft ← nleft - 1

If nleft ≠ Ø go to step TALLY
else to step NOCAND

Keep - This step is entered from step 9 under the conditions

logr ≤ Group # < higr - AND-

MNSN ≤ spud ≤ MXSN

Do: A ← X-Reg copy of Cand. List entry made in step winnow.

Store Cand. List entry at (stpt)

stpt ← stpt + 1

Tally - This step is entered from steps

CANCEL - nleft ≠ Ø

KEEP - Unconditionally

Do: ncand ← ncand - 1

If ncand ≠ Ø go to step winnow - else next

; 10 - (Out-of-Loop here to End)

Store nleft in left byte of Cand. List header word at address (clad)

; 11 - Bump return address to call +2:

rtad ← rtad + 1

Abnorm - Entered from Step 11 (Returning to Call +2)
Or from Step 13 (Returning to Call +1)

Clean up stack, i.e., return to available status the 8 locations on stack that were appropriated for temporary local storage in Steps 1 - 5 inclusive.

S-Reg ← S-Reg + 8

Restore X, E, B and A-Reg's from stack

Return

NOCAND - This step is entered from step CANCEL when nleft = Ø, i.e., the Candidate List has been entirely eliminated.

B-Reg ← adef = ETF + 16 * EFN (Step 3)

Replace ETF identity field (EIDD relative to B-Reg) by code NOFA2 (Using mask EIDM).

; 12 - Replace ETF display code field (EDISD Rel B) by code UNKNO (Using mask EDISM)

; 13 - Clear left byte of Cand. List header word (ADDR (clad)).

Go to step ABNORM

3.1.9.2.1 E2ADR - Subroutine of ANLV2
Compute EL2 Address

The subroutine is also called by TRNSL8 in Emitter Classification 1 (Document No. 53959-GT-0760) and by ANAMB in Analysis Return Subroutine ANEC3.

On input the right byte of A-Reg contains an Index (Ground Number/ Emitter Library No.). On output the left byte is cleared and

B-Reg = EL2 + 11 * (Index -1)

E2ADR - Mask out A-Reg left byte

; 1 - Copy A to B

; 2 - Multiply B-Reg by 11. This is done by a sequence of double - B's ($B \leftarrow B + B$) and add's ($B \leftarrow B + A$) that is much faster than an MPY instruction.

; 3 - Add the constant EL2 - 11 to B

Return

3.1.9.3 ANEL1 - Subroutine of ANEC2
New Emitter Link Analysis - 1

This version is a dummy. Its one and only step is a return to Call +1 indicating: No Contemp. analysis wanted.

Any non-dummy version must preserve and restore the A and B-Reg's.

3.1.10 ANEC3

ANEC3 shall be entered with X-Reg pointing to a two word block such that the right byte of word 1 contains the emitter track file # (EFN) and word 2 = the Candidate List Address (CLAD). This value of X-Reg shall remain as the entry value of X-Reg for each of the subroutines called by ANEC3.

ANEC3 shall consist only of the following steps:

- Call New Emitter Link Analysis #2 (ANEL2)
- Call Family Associateion (ANFAM)
- Call Ambiguity Resolution (ANAMB)
- Bump return address to call + 2 to cause a No-Analysis return to Analysis Return Driver.
- Return

3.1.10.1 ANEL2 - Subroutine of ANEC2
New Emitter Link Analysis #2

ANEL2 shall be entered with X-Reg set as upon entry to ANEC3.

The present version of ANEL2 shall be a dummy in that it shall only cause the platform link field of ETF (EFN) to point to itself, i.e., contain the value EFN. The steps shall be as follows:

ANEL2 - Save A and B-Reg's on stack

; 1 - $A \leftarrow ((X)) = \begin{cases} \text{Junk in left byte} \\ \text{EFN in right byte} \end{cases}$

Call SOGET: $\begin{cases} B \leftarrow \text{ETF} + 16 * \text{EFN} \\ A \leftarrow \text{EFN, left byte cleared.} \end{cases}$

Save EFN, now in A-Reg, on stack

; 2 - Fetch ETF word containing platform link field. This shall be at displacement EPLKD relative to B-Reg. This shall put in A-Reg a byte split word whose left byte is to be retained and whose right byte is to be replaced by EFN.

Mask out A-Reg. right byte and OR in EFN from top-of-stack ($S \leftarrow S + 1$).

Store result at location from which fetch was made at beginning of this step.

; 3 - Restore B and A-Reg's from stack

Return

3.1.10.2 ANFAM - Subroutine of ANEC 3
Family Association

The present version of ANFAM shall be a dummy. Its one step shall be an Exit instruction.

3.1.10.3 ANAMB - Subroutine of ANEC3
Ambiguity Resolution

Ambiguity resolution shall be entered with X-Reg set as upon entry to ANEC3. ANAMB shall reduce the designated candidate list to a single entry (The winner, hereafter) by selecting the candidate whose weighting factor is highest, breaking ties, if any, in favor of the lower-numbered candidate.

ANAMB shall set various ETF fields with information taken from the winner's EL2 file as detailed below.

ANAMB shall send a classification-concluded (update) message to the Executive.

The steps followed by ANAMB shall be the following:

ANAMB - Save A, B, E, X-Reg's on stack

; 1 - Set X = Candidate List Address

; 2 - Set A = Candidate List Header word

$= ((X))$ and $X \leftarrow X + 1$

The header word shall consist of NCAND (Candidate List Length) in the left byte and EFN in the right byte.

The bytes shall be separated by calling an internal subroutine Bunpak so that

$A \leftarrow EFN$
$E \leftarrow NCAND$

EFN shall now be stored in the third word of the update message (UPEFN).

NCAND shall be pushed to stack and referred to hence by S-relative instructions with symbolic displacement NCAND. In the following text, we refer to contents of said location as ncand.

; 3 - External subroutine SOGET (in Sorter Message Processing - document 53959-GT-0755) shall now be called using A-Reg = EFN as input to do.

$$B \leftarrow ETF + 16 * EFN$$

This value shall be pushed to stack and value thus stored shall be refered to by S-relative instructions with symbolic displacement ADEF; contents referred to in following text as adef.

; 4 - E-Reg shall be set = -1 as initial value of highest weight to be carried throughout forthcoming loop in said register

Room shall be made on stack for a temporary location for storing the Candidate List entry of candidates as successive maximal weighting factors are discovered in the following loop. (S-Relative symbolic displacement: Winner, contents: winner).

AMLOOP - This step shall be entered 1st time from step 4 and N-1 thereafter from step tally, where N = original value of ncand as set in step 2.

This step shall do:

Pick up next Candidate List item = ((X)); X $\leftarrow$ X + 1

This shall load A-Reg with Candidate identity code (left byte) and group # (right byte).

Call External Subroutine ELADR (part of ANEC2) which shall mask out A-Reg left byte and set

$$B \leftarrow EL2 + 11 \cdot (\text{Group \# } -1)$$

; 5 - Fetch Candidate's weighting factor word which shall be at displacement MFCTD relative to B and mask out extraneous fields.

; 6 - Candidate weight shall be compared to current maximum (in E-Reg):

Cand. Wt $\leq$ Current Max: Go to step tally
else next

; 7 - Replace current max by Candidate Weight

Fetch Candidate List entry, which shall be found at -1 relative to X and store it as current winner.

Tally - This step shall be entered from either

Step 6 - Candidate weight $\leq$ Current maximum

Step 7 - Candidate has become new winner.

This step shall test for loop completion by doing:

ncand $\leftarrow$ ncand - 1

If ncand still > Ø go to step AMLOOP else next

; 8 - Loop is now complete. E-contains maximum Candidate weight (not used) and the winning candidate's Cand. List entry is stored as winner.

This step shall do:

X $\leftarrow$ adef = ETF + 16 · EFN

A $\leftarrow$ winner = Byte-split word
= (Ident, Group #)

Internal byte unpacking subroutine BUNPAK shall be called to do:

A $\leftarrow$ Group #

E $\leftarrow$ Ident

; 9 - Winner Group # shall be saved in B freeing A for use in this and next three steps.

This step shall insert winner Group # in ETF ELN field:

$A \leftarrow (ELND + X)$

$A \leftarrow A \wedge Mask$; Clears ELN field

$A \leftarrow A \vee B$; Inserts Group #

$(ELND + X) \leftarrow A$

; 1Ø - This step shall insert winner ident in ETF ident field:

$A \leftarrow (EIDD + X)$

$A \leftarrow A \wedge Mask$; Clears EID field

$A \leftarrow A \vee E$; Inserts Ident

$(EIDD + X) \leftarrow A$

; 11 - Winner ident (8-Bit field) shall now be compared to 16 to see whether its particular value will fit into the 4-Bit ETF Display code field.

If no: Clear E-Reg

; 12 - This step shall store the 4 least significant bits of E-Reg in ETF Display code field:

$A \leftarrow (EDISD + X)$

$A \leftarrow A \wedge Mask$; Clears EDIS field

$E \leftarrow$ Left Shift (E); Appropriate # bits to align

$A \leftarrow A \vee E$

$(EDISD + X) \leftarrow A$

; 13 - This step shall put Winner Group # (saved in B) back into A as input to

Call E2ADR (See Step AMLOOP) so that

$B \leftarrow EL2 + 11 * (Group \# -1)$

; 13 -continued- The balance of this step shall use B as just set to fetch and isolate Winner's EL2 platform code and test it against standard "Naval" code.

= : E ← 1 in ETF ENAV Bit position

≠ : E ← Ø

; 14 - This step shall set or reset ETF ENAV bit depending on result of Step 13.

A ← (ENAVD + X)

A ← A ∨ E

(ENAVD + X) ← A

; 15 - This step shall send an update message to the Executive:

X ← Address UPMSG

Call EXMES

; 16 - This step shall clear up stack by

S ← S + 3

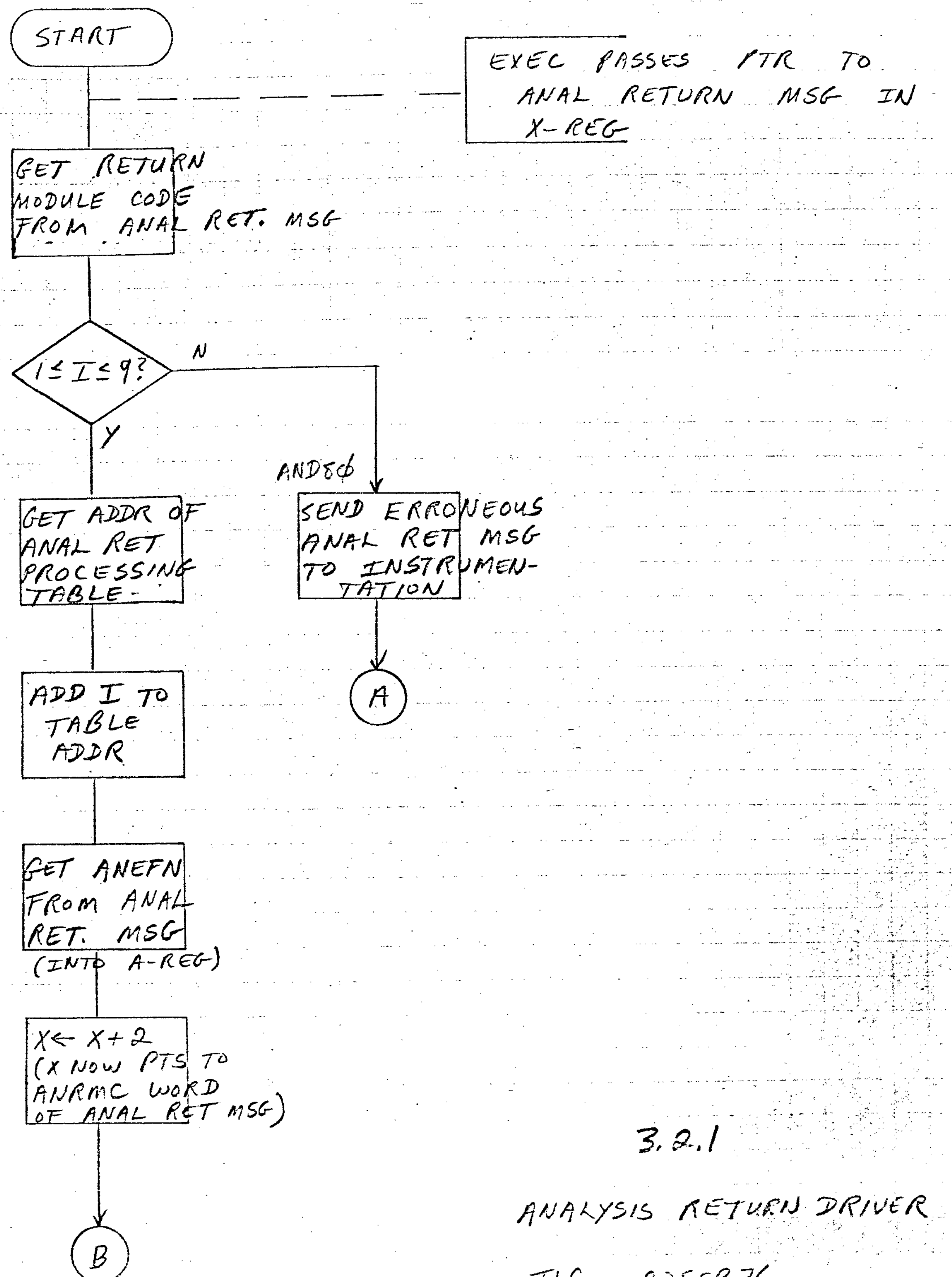
Restore X, E, B, A-Reg's from stack

Return

3.2 SUBPROGRAM FLOW DIAGRAMS

The logic flow for all routine comprising this subprogram is shown in the following flow diagrams. The flow diagrams are labeled so as to correspond to paragraph 3.1. That is, flow diagram 3.2.9 is described in paragraph 3.1.9. Data extraction points for instrumentation are shown as comment blocks with the text "DP____".

ANDR

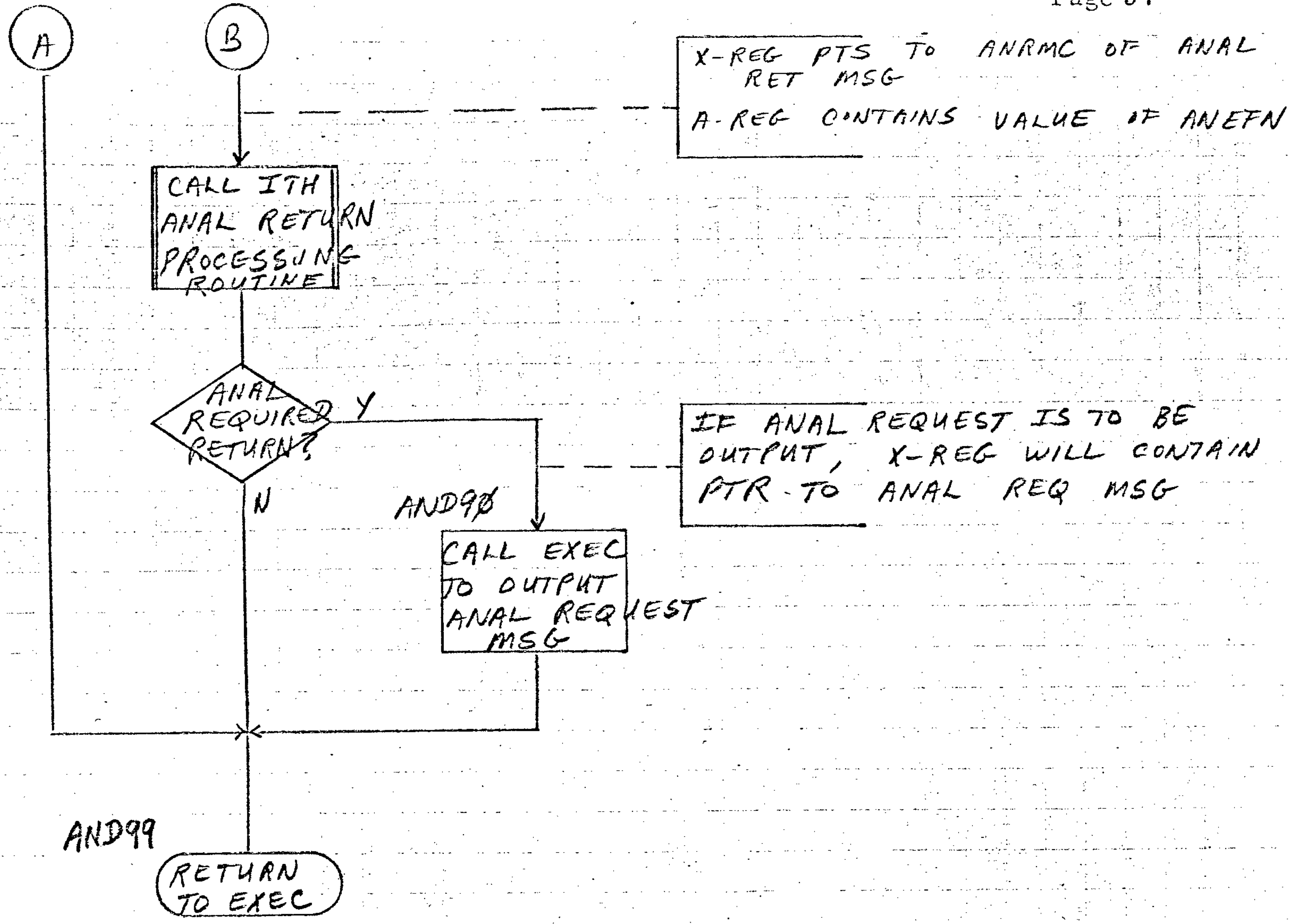


3.2.1

ANALYSIS RETURN DRIVER

TLC 20SEP76

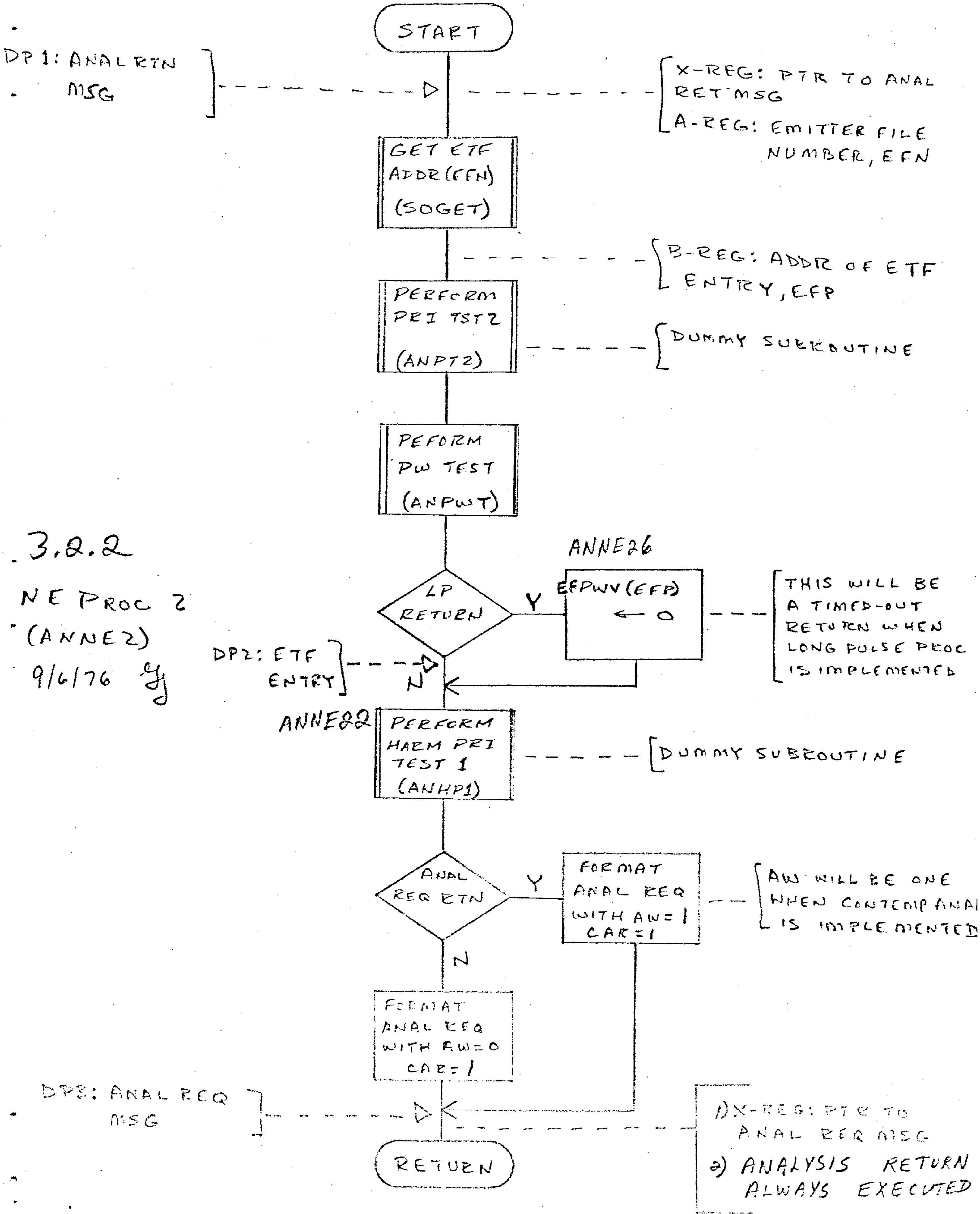
SHT 1 OF 2

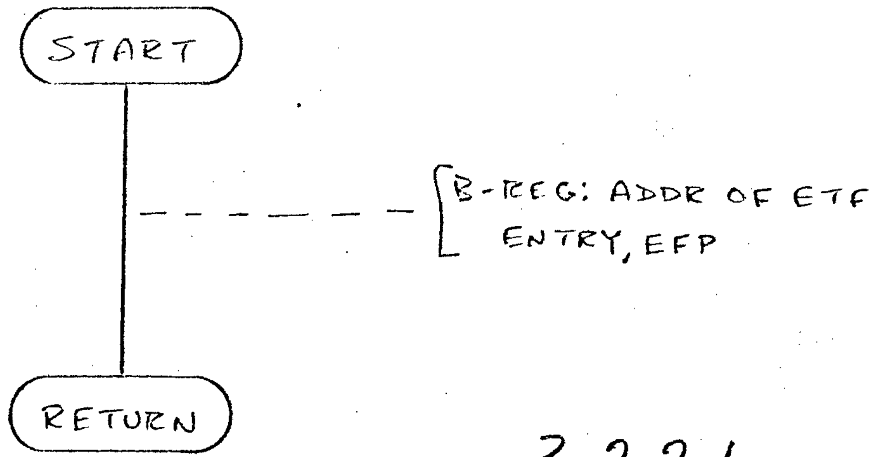


3.2.1 (Concluded)

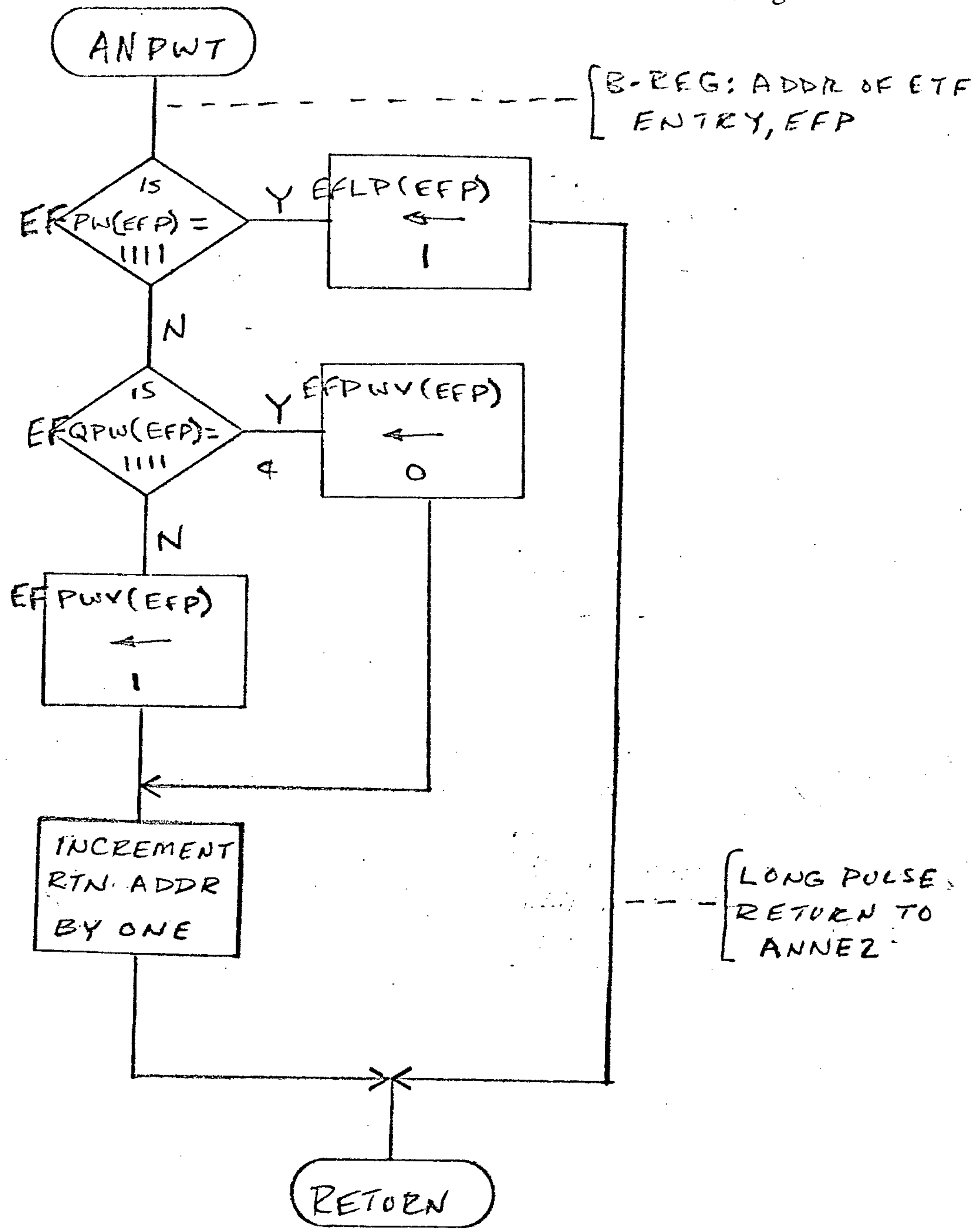
ANALYSIS RETURN DRIVER
JLC 20 SEP 76
SHT 2 OF 2

3.2.2
NE PROC 2
(ANNE2)
9/6/76





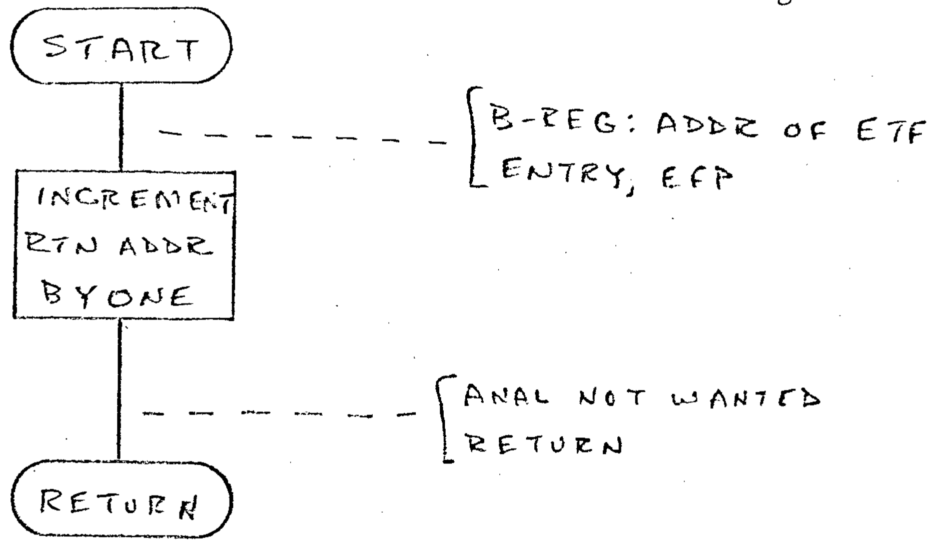
3.2.2.1
PRI TEST 2
(ANPT2)
9/6/76 GJ



3.2.2.2

PW TEST (ANPWT)

~~8/19/76~~

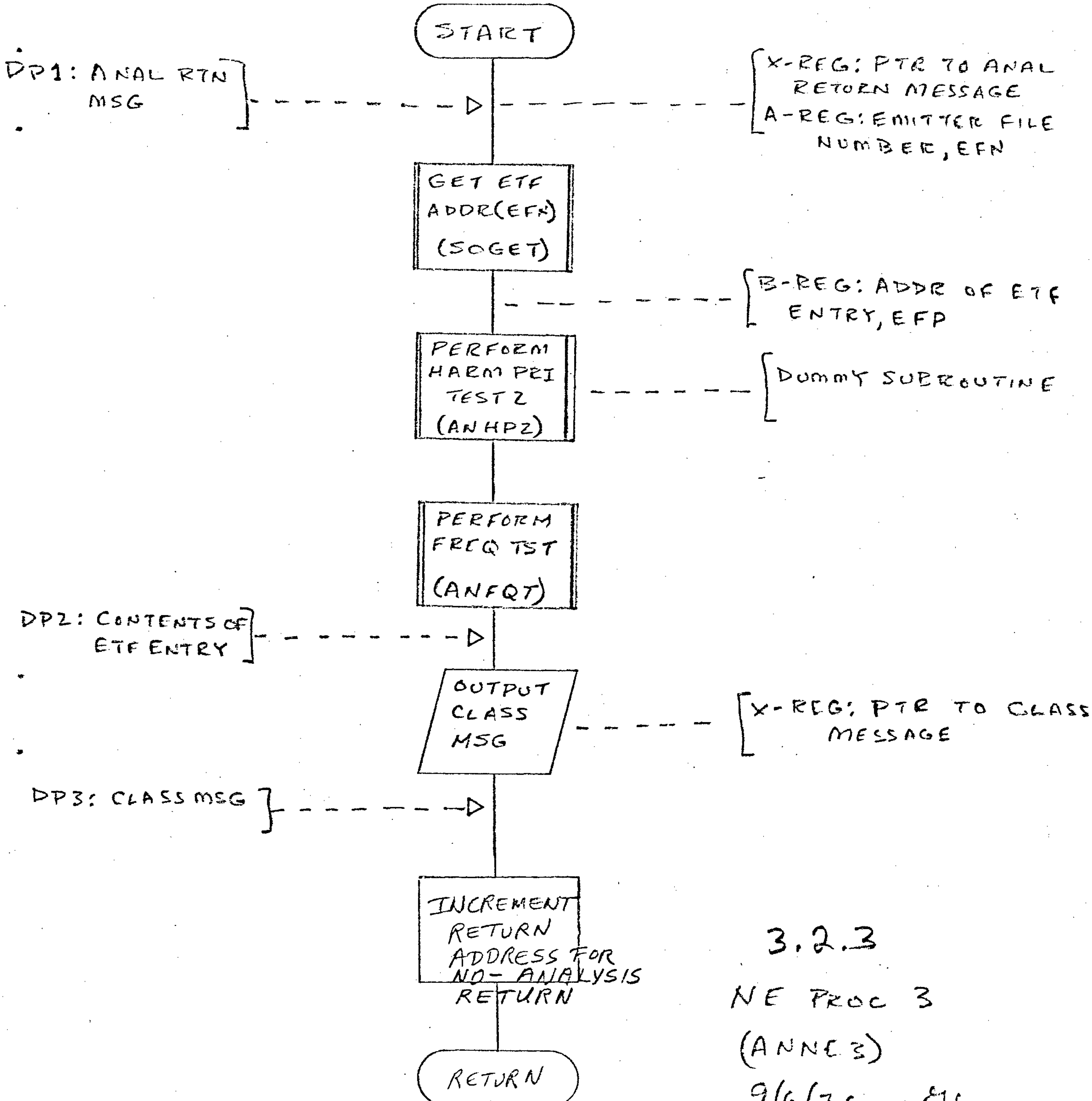


3.2.2.3

HARMONIC PRI TEST 1

(ANHP1)

9/6/76

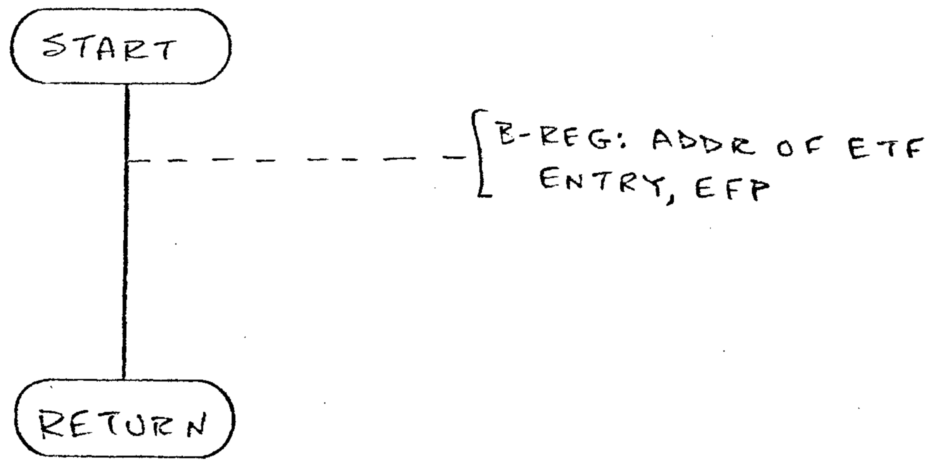


3.2.3

NE PROC 3

(ANNC3)

9/6/76

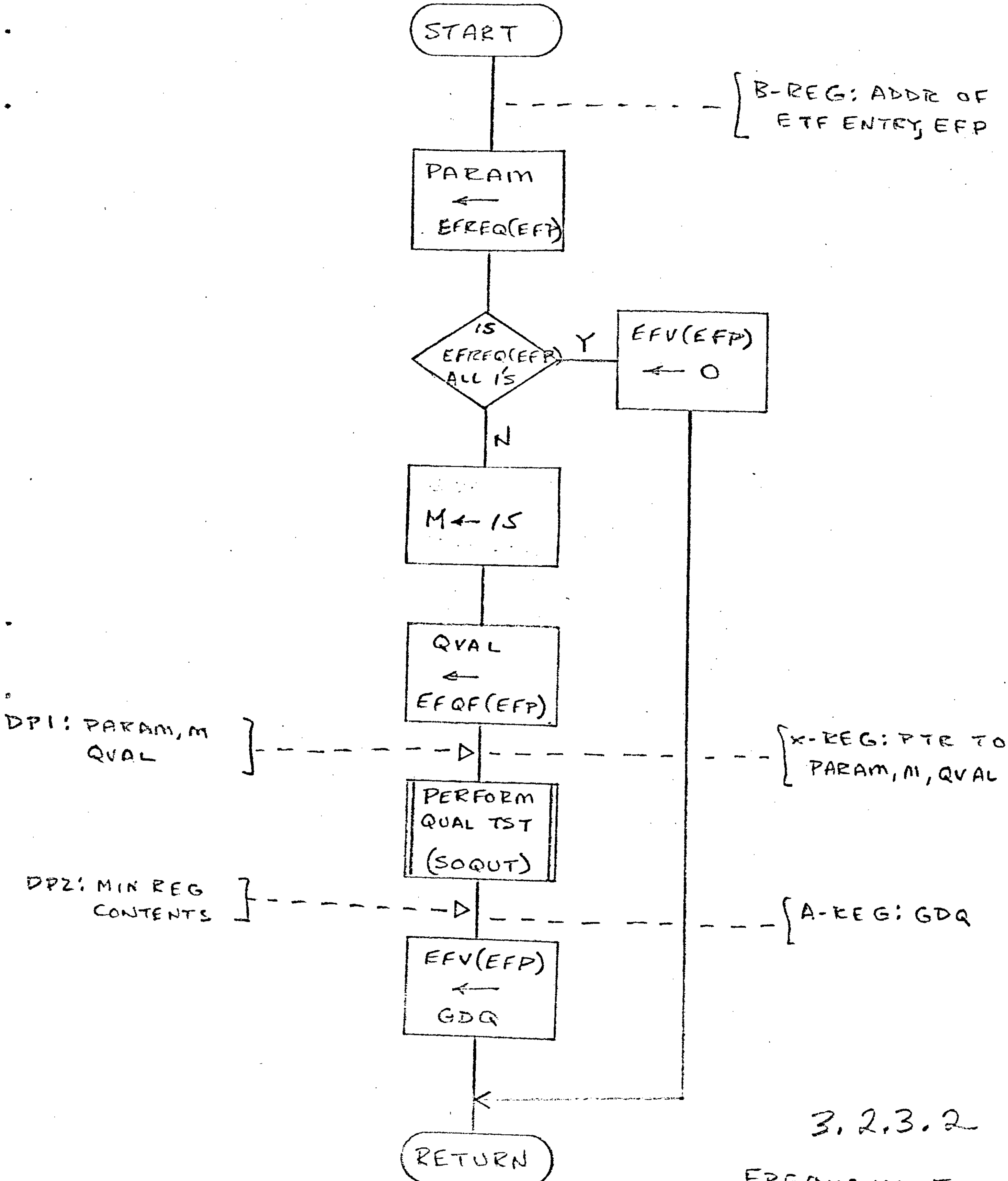


3.2.3.1

HARMONIC PRI TEST 2

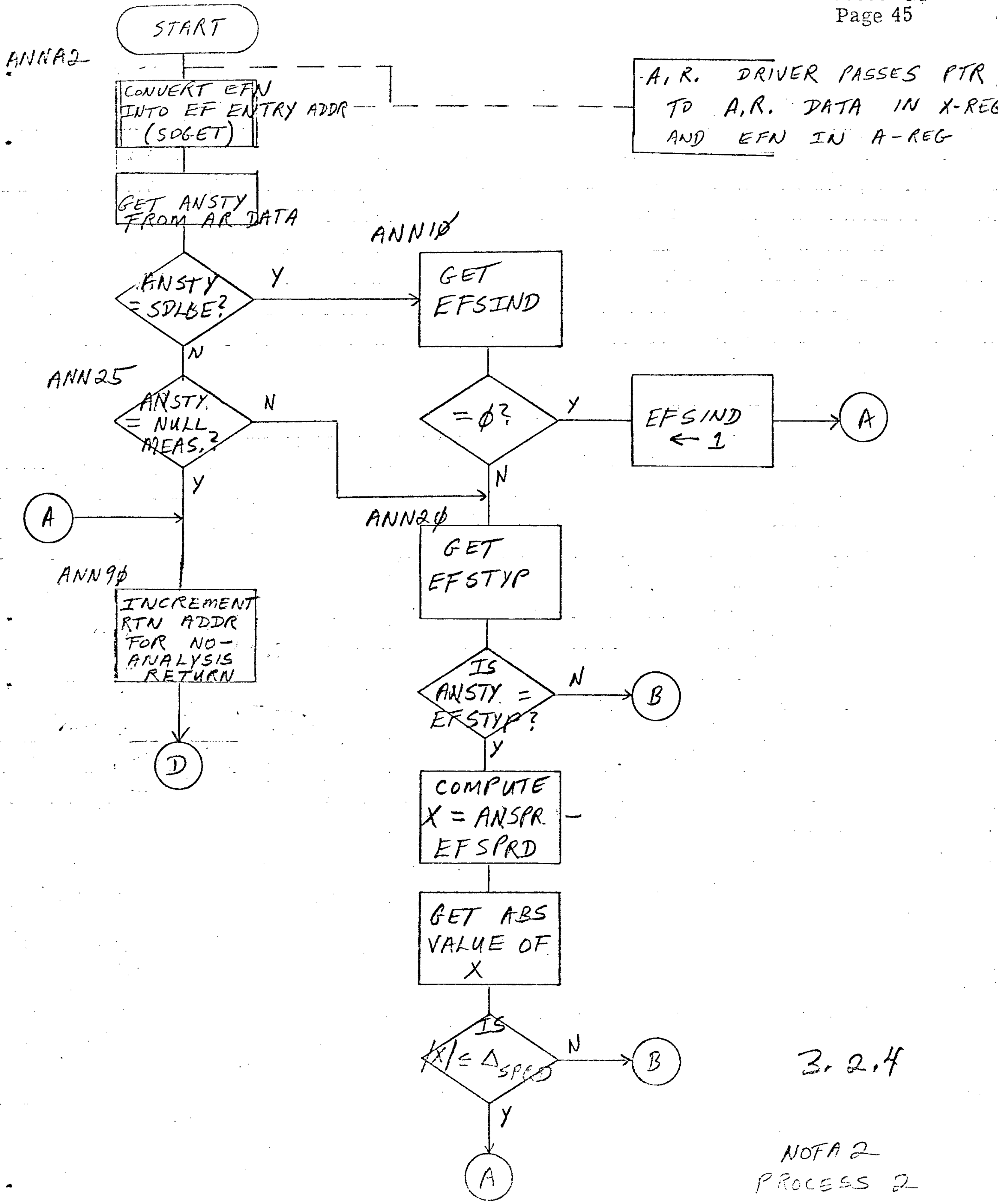
(AN HP2)

9/6/76



3.2.3.2

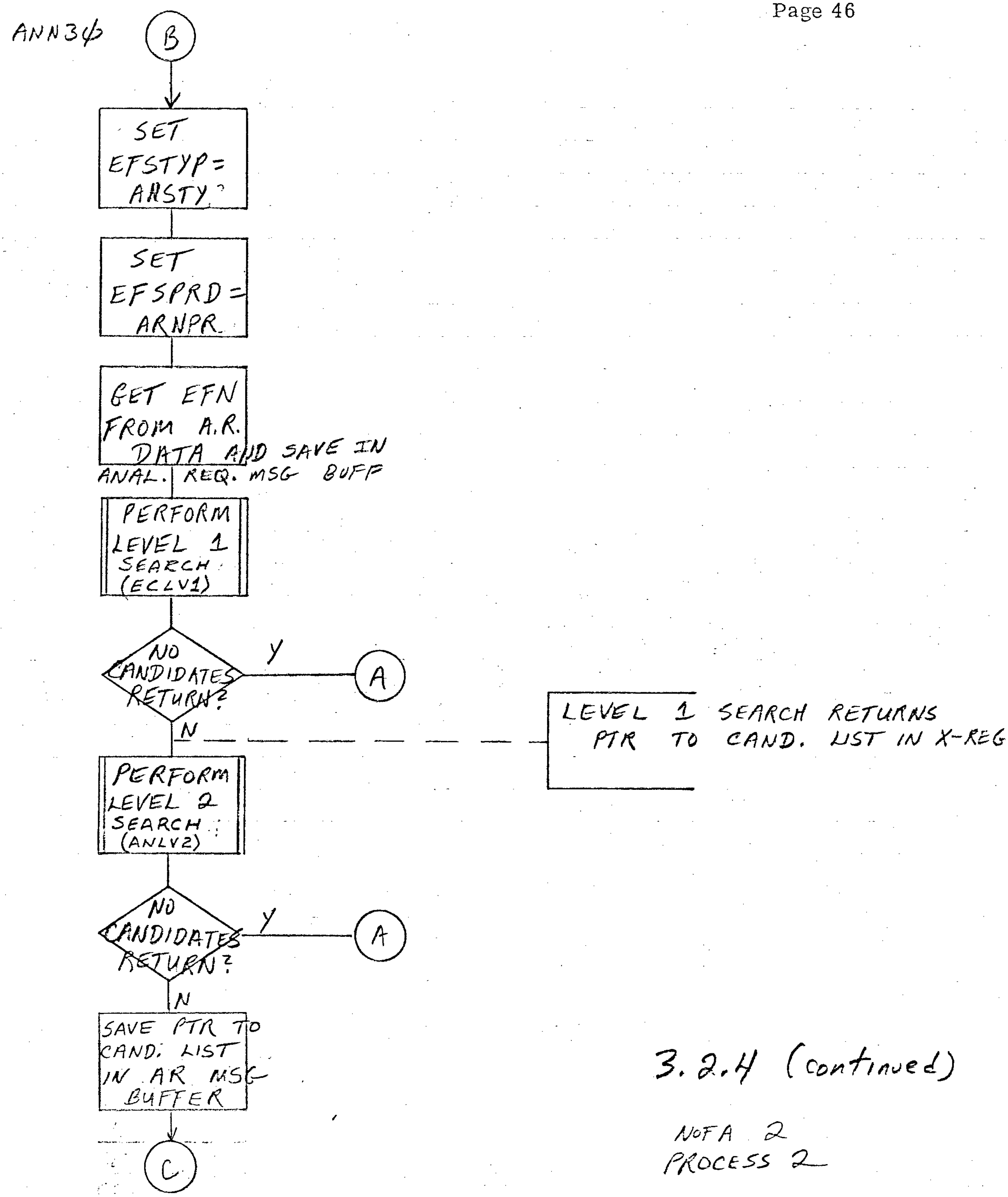
FREQUENCY TEST
(ANFQT)
9/6/76



3.2.4

NOFA 2
PROCESS 2

TLC 27 AUG 76
SHT 1 OF 3

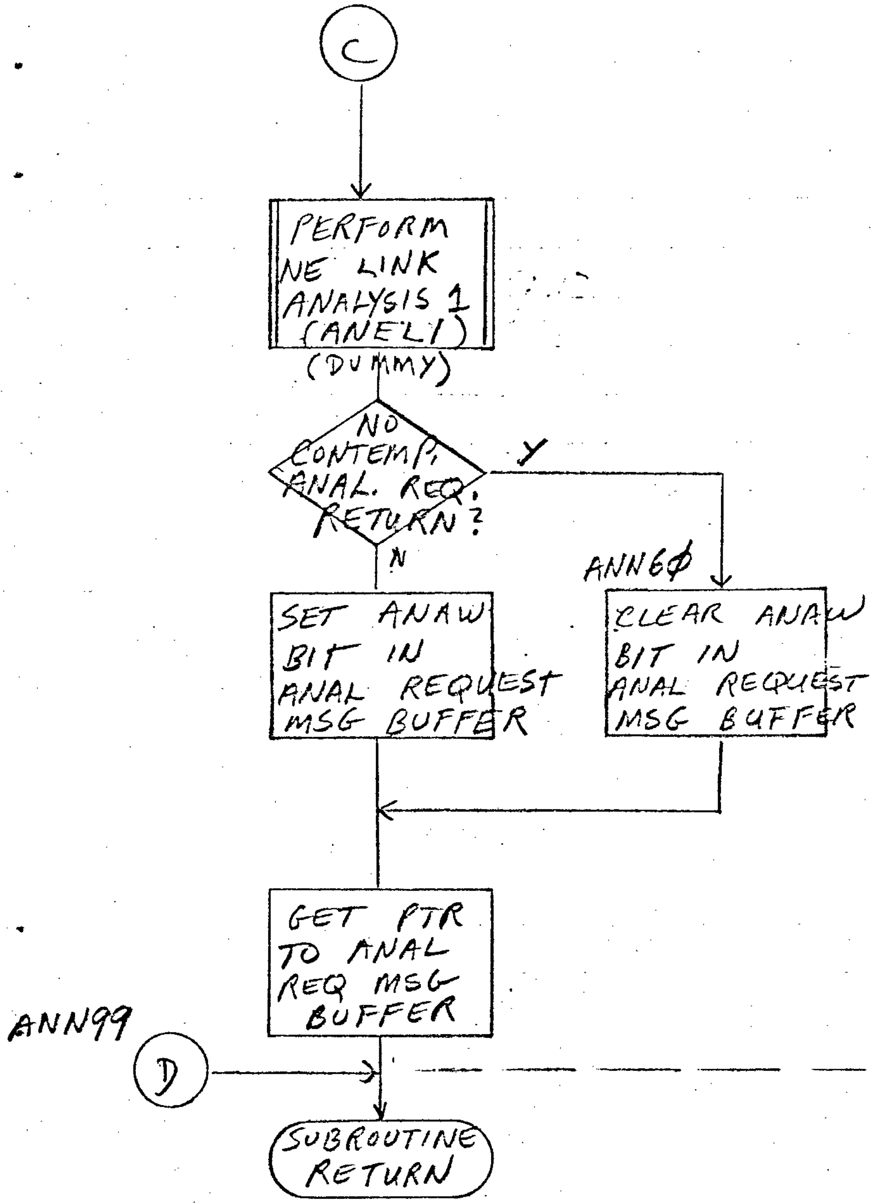


3.2.4 (continued)

NOFA 2
PROCESS 2

TLC 27 AUG 76

SHT 2 OF 3



NOFA2 PROCESS 2 HAS 2 RETURNS TO AR DRIVER :

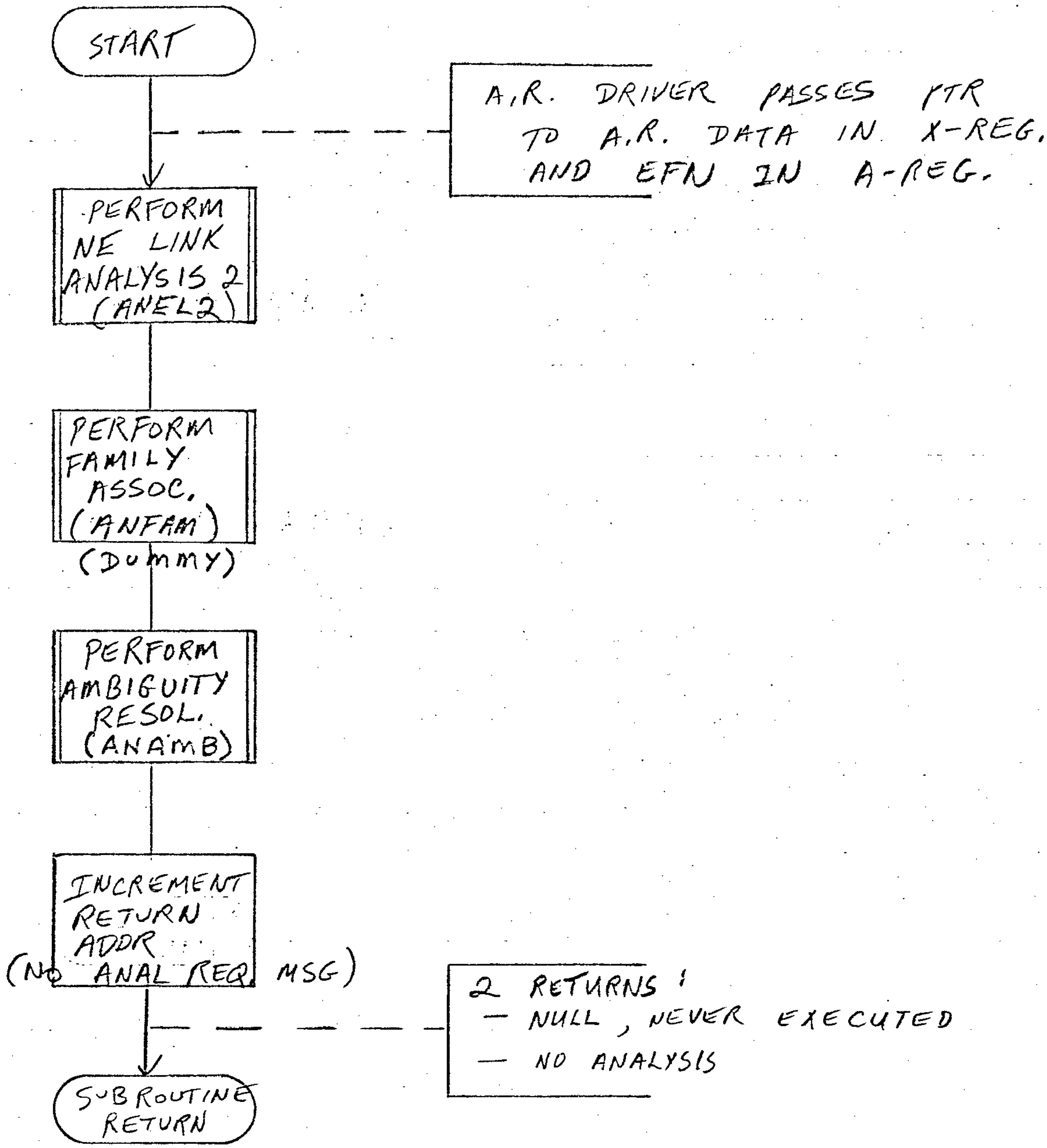
1. OUTPUT "ANAL REQ" MSG
2. OUTPUT NO ANAL MSGS

IF NO. 1., X HAS PTR TO ANAL REQ MSG BUFFER

3.2.4 (concluded)

NOFA 2
PROCESS 2
TLC 30 AUG 76
SHT 3 OF 3

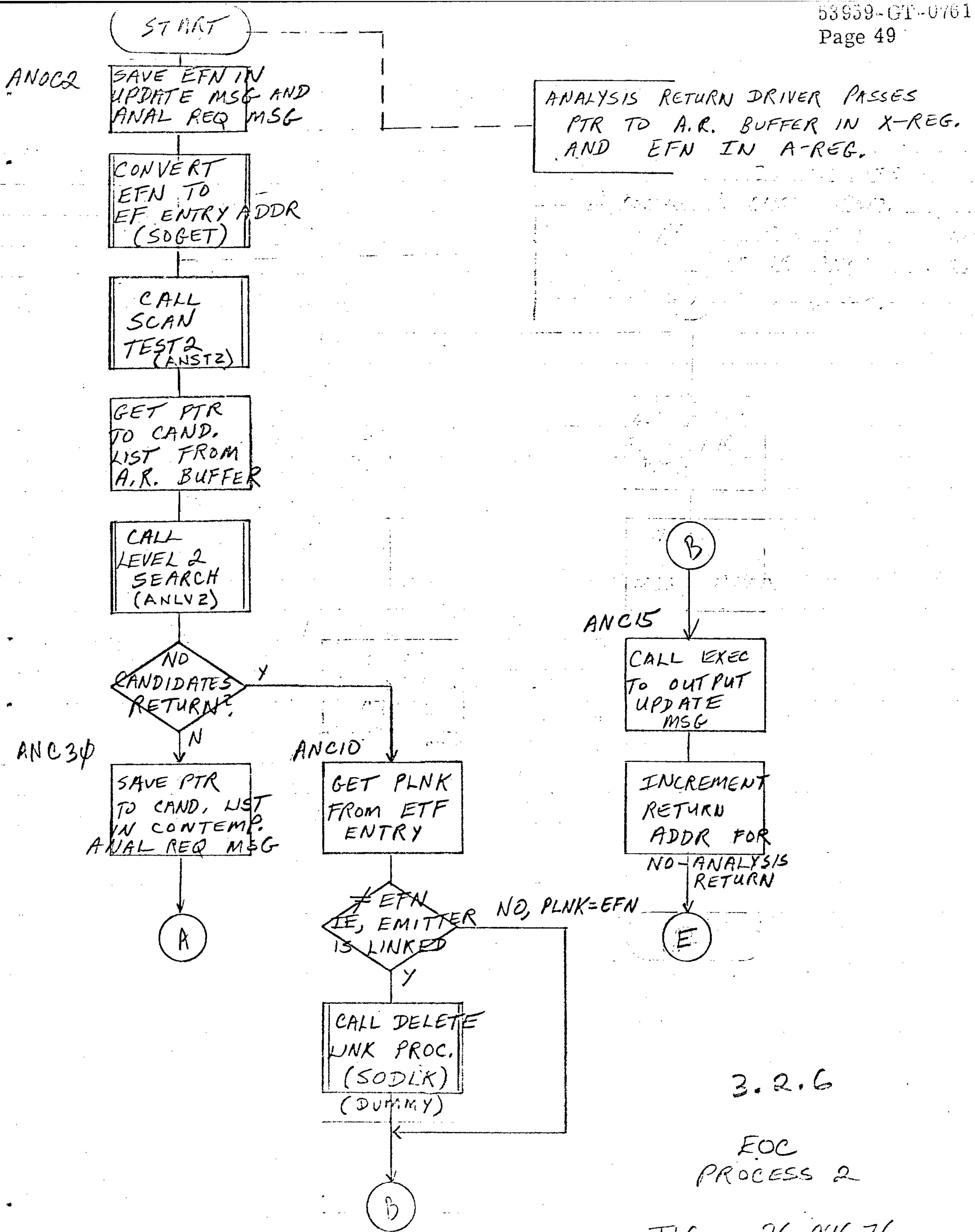
ANNA3
ANDC3



3.2.5 and 3.2.7

NOFA2 PROCESS 3
OR
EOC PROCESS 3

TLC 27AUG76

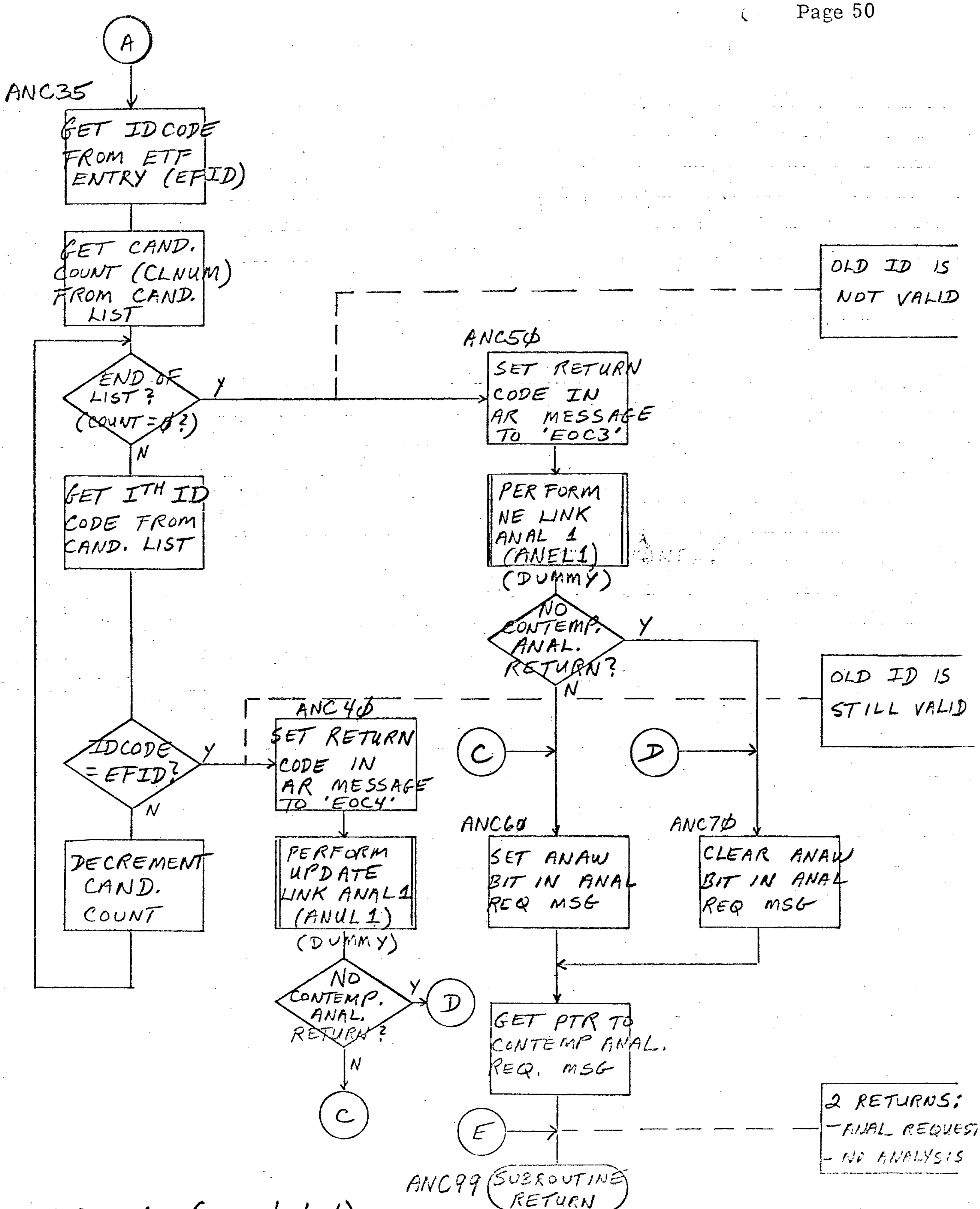


3.2.6

EOC PROCESS 2

TLC 26 AUG 76

sht 1 of 2

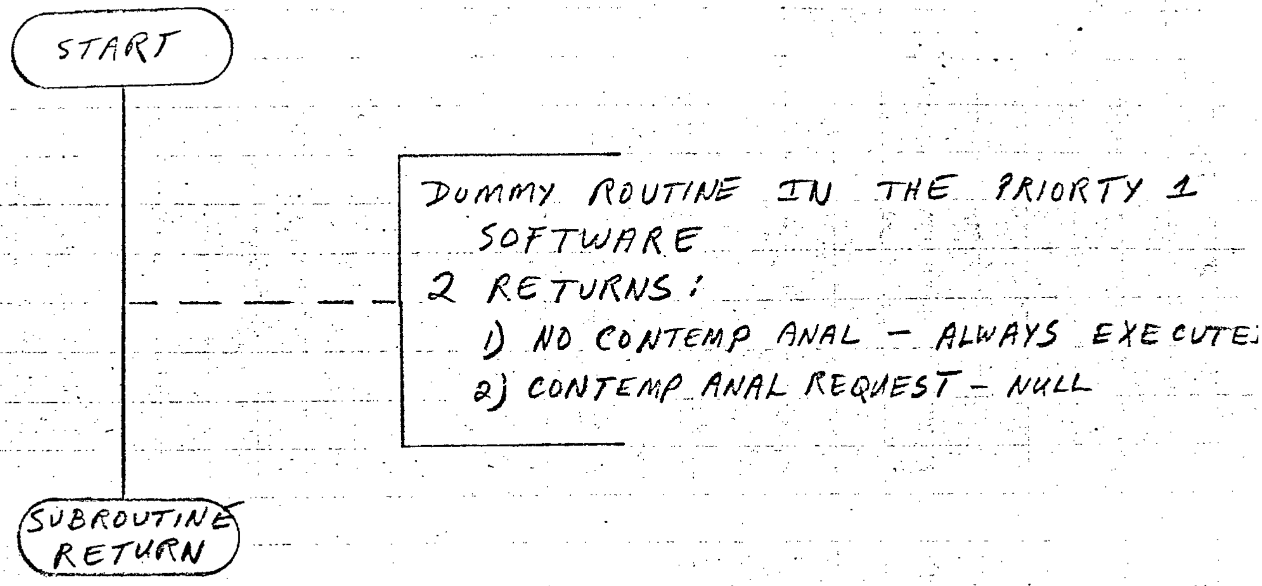


3.2.6 (concluded)

EOC PROCESS2
TLC 26 AUG 76

SHT 2 OF 2

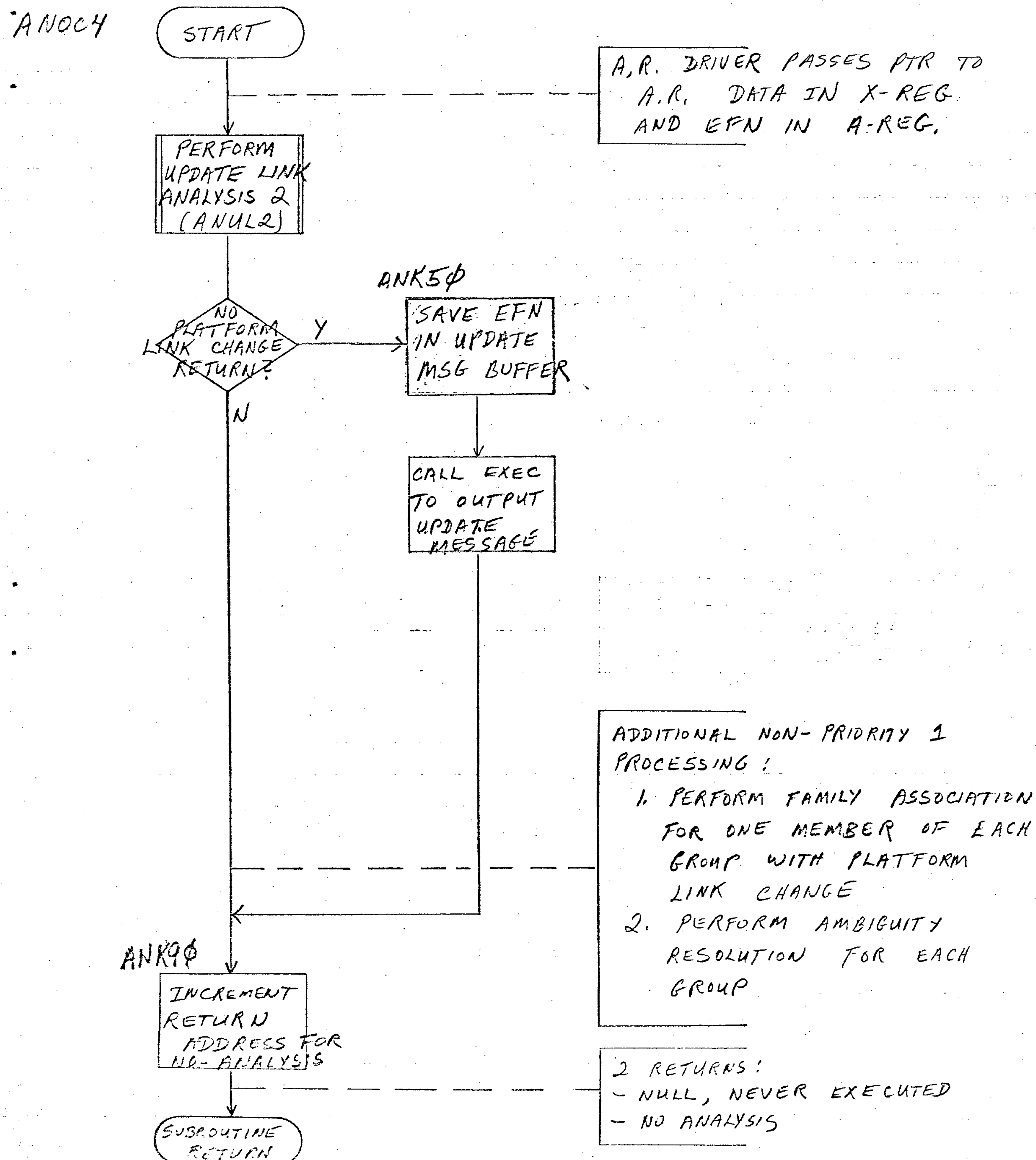
ANUL1



3.2.6.6

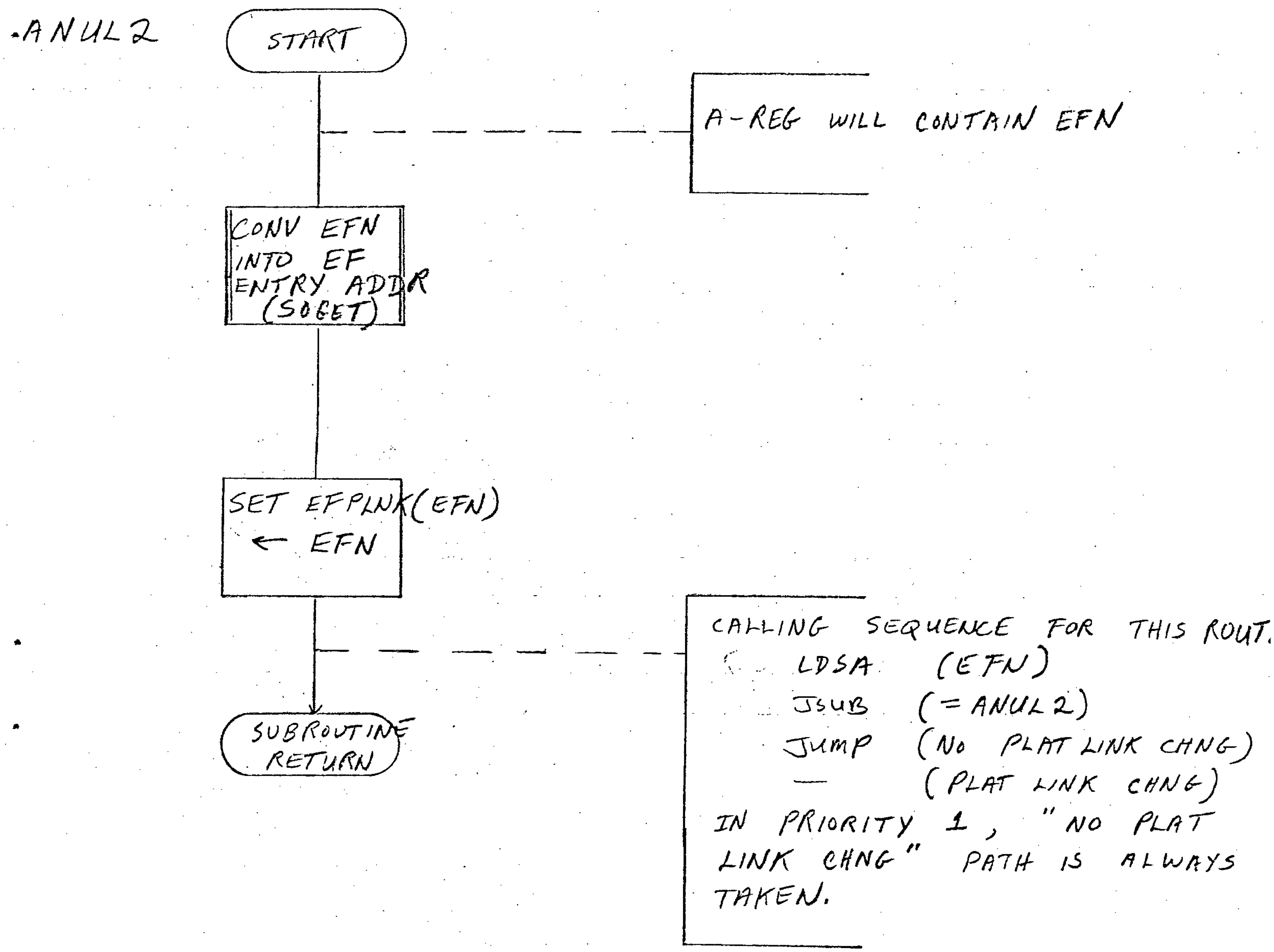
UPDATE LINK ANALYSIS 1

TLC 19 OCT 76



EOC PROCESS 4

3.2.5 TLC 30 AUG 76



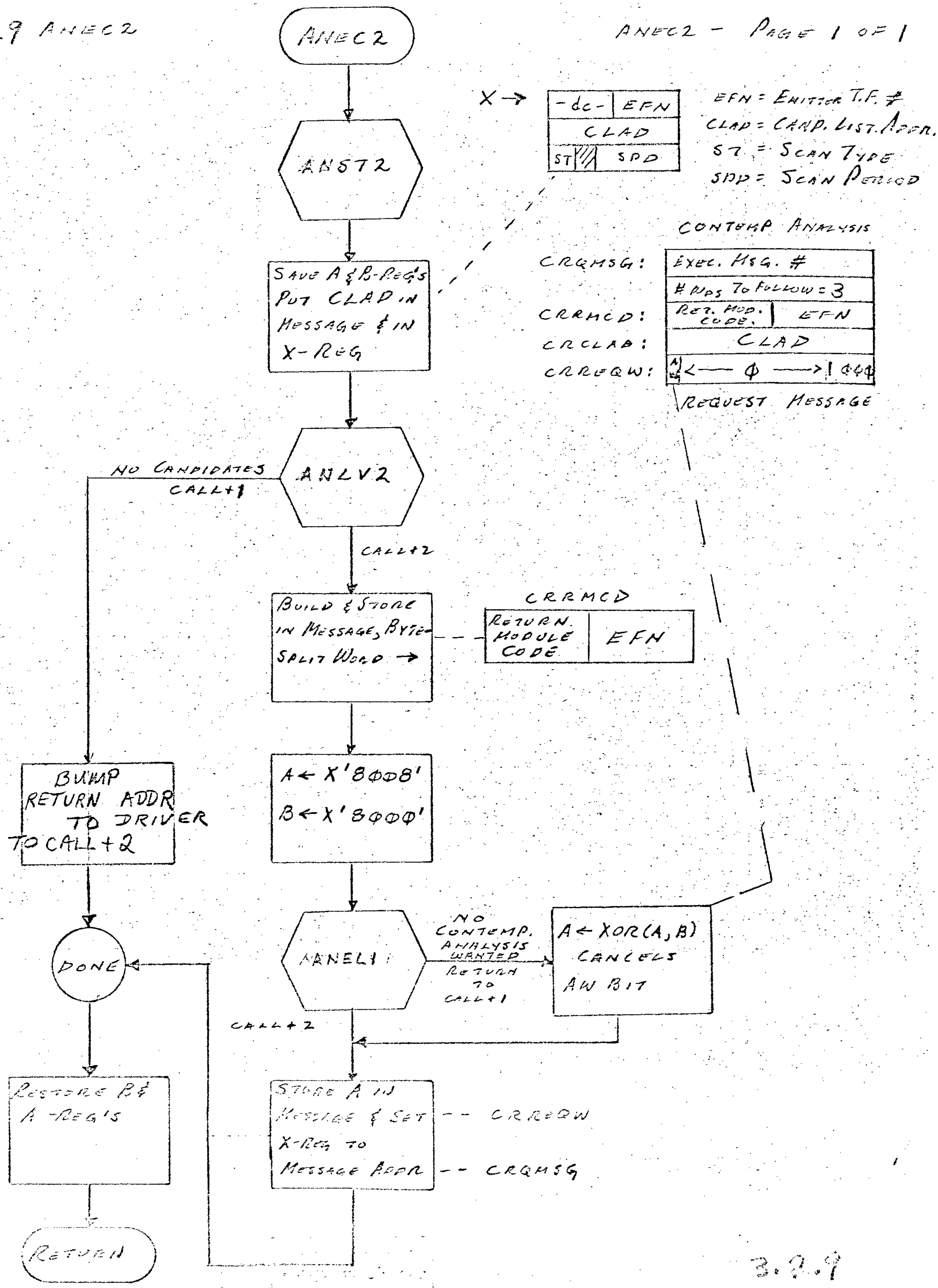
3.2.8.1

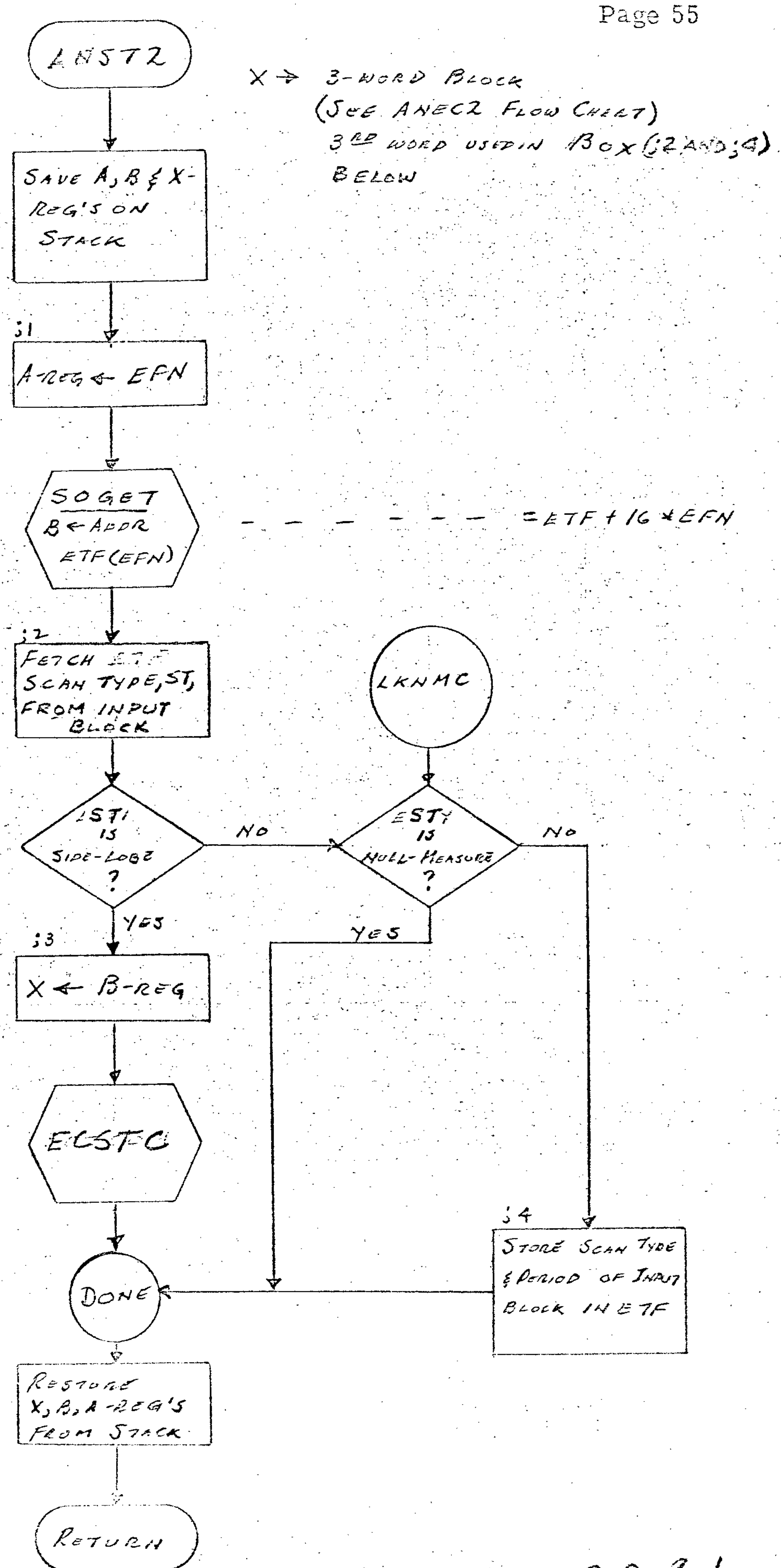
UPDATE LINK ANALYSIS 2

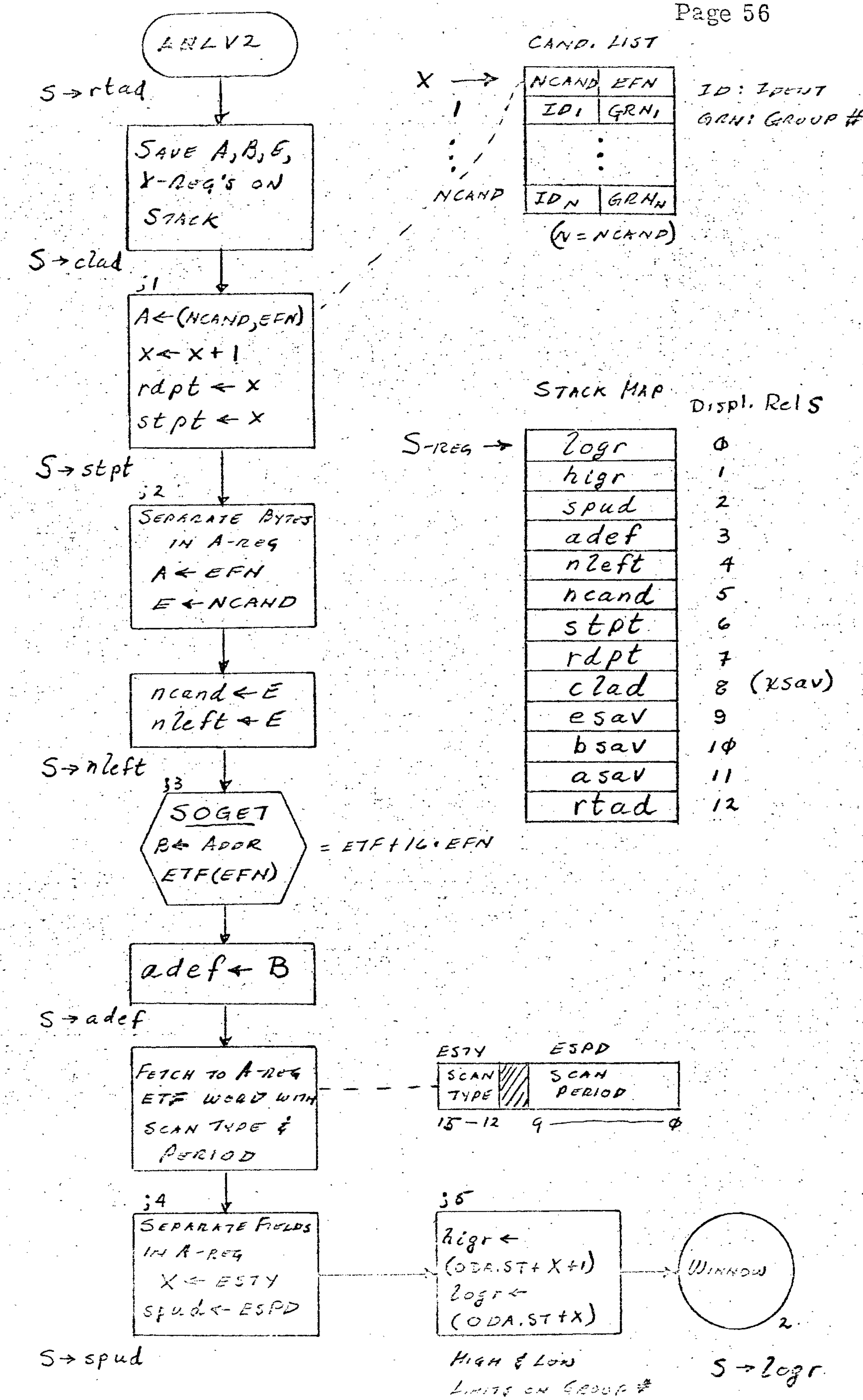
TLC 30 AUG 76

3.2.9 ANEC2

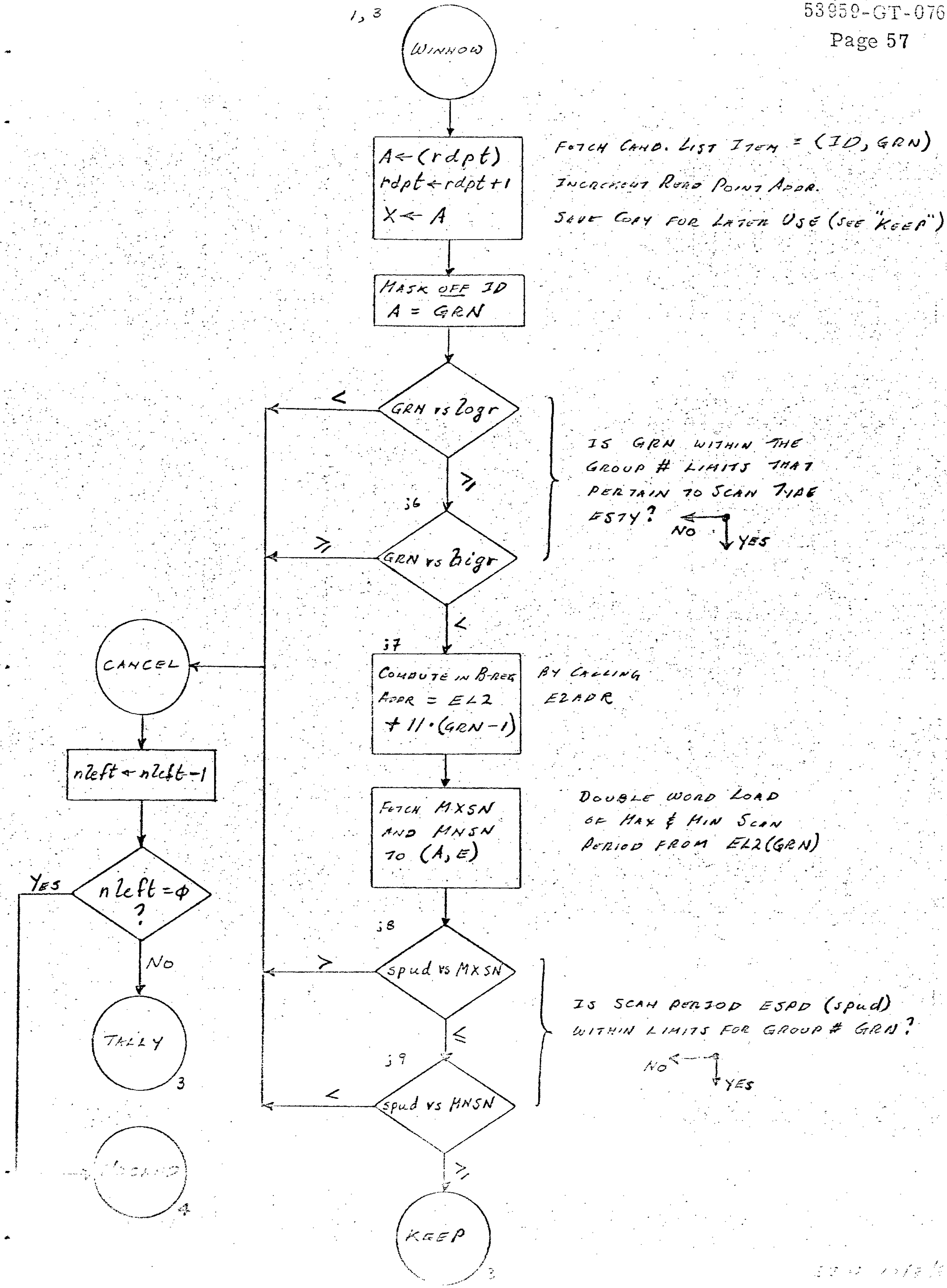
ANEC2 - PAGE 1 OF 1

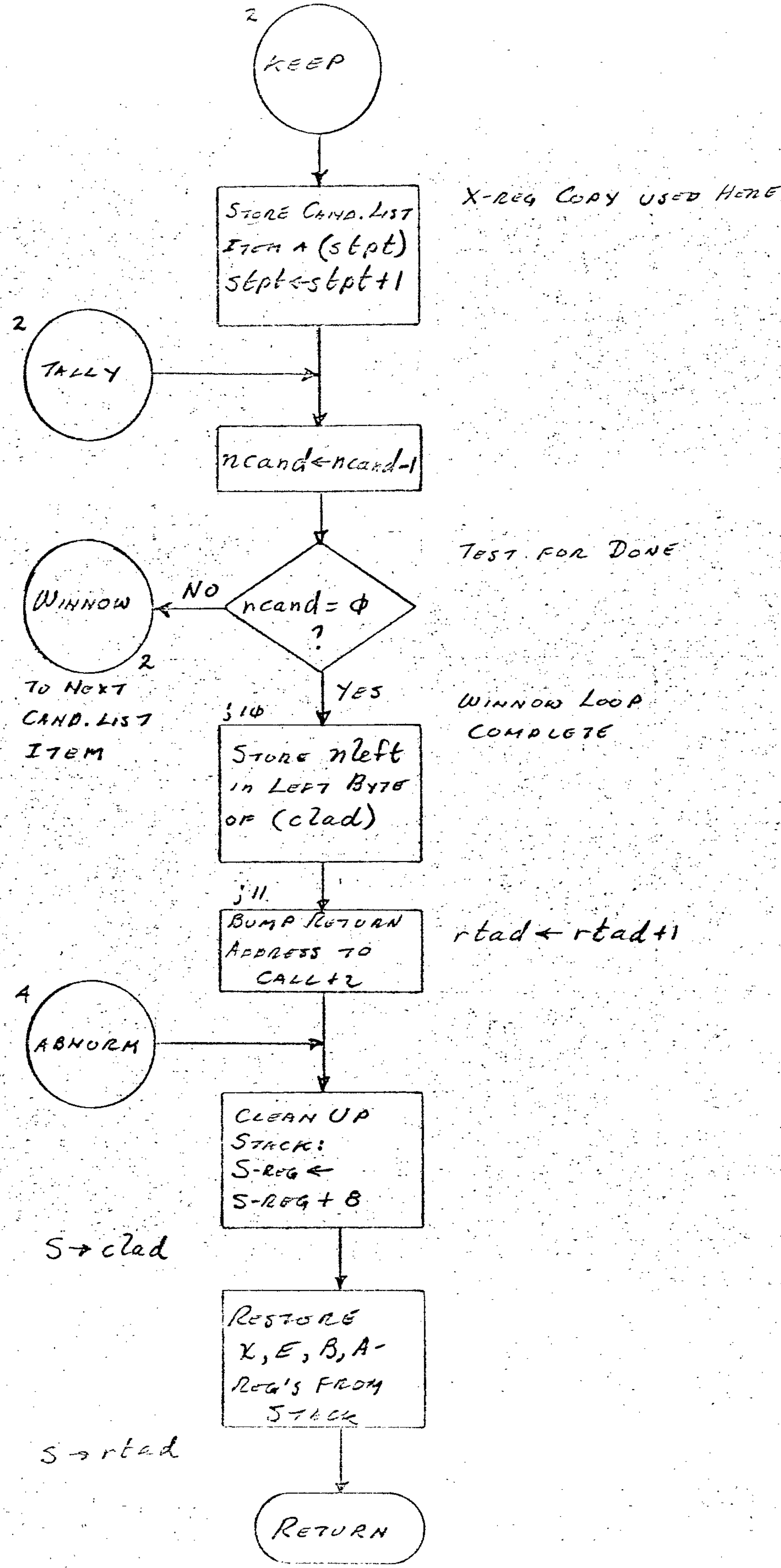


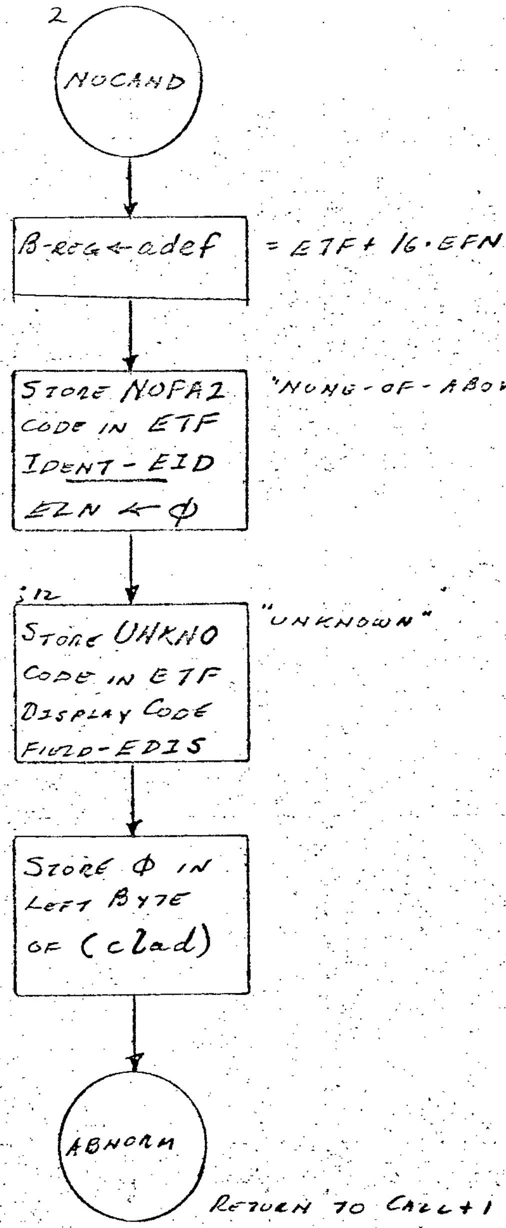

ANST2
X → 3-WORD BLOCK
(SEE ANEC2 FLOW CHART)
3RD WORD USED IN BOX (;2 AND ;4)
BELOW
SAVE A, B & X-REG'S ON STACK
;1
A-REG ← EFN
SOGET
B ← ADDR ETF(EFN)
= ETF + 16 * EFN
;2
FETCH ETF SCAN TYPE, ST, FROM INPUT BLOCK
LKNMC
ST IS SIDE-LOBE ?
NO
ST IS NULL-MEASURE ?
NO
YES
YES
;3
X ← B-REG
ECSFC
;4
STORE SCAN TYPE & PERIOD OF INPUT BLOCK IN ETF
DONE
RESTORE X, B, A-REG'S FROM STACK
RETURN

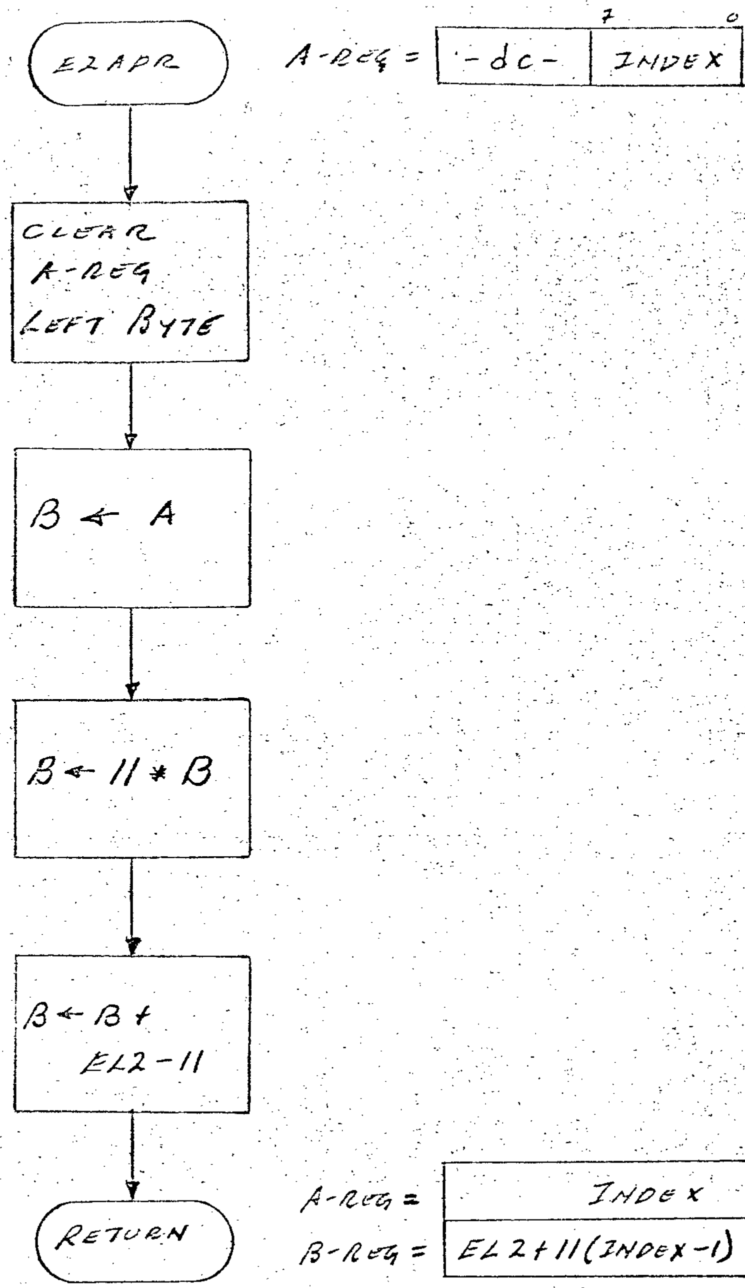


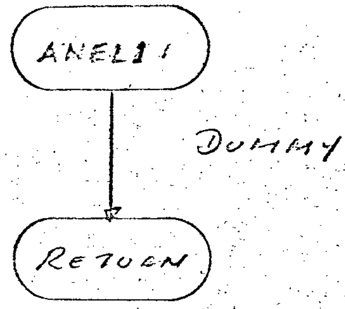
ALL STACK MAP BUILDING OPERATIONS ARE ON THIS PAGE


1, 3
WINNOW
A ← (rdpt)
rdpt ← rdpt + 1
X ← A
FETCH CAND. LIST ITEM = (ID, GRN)
INCREMENT READ POINT ADDR.
SAVE COPY FOR LATER USE (SEE "KEEP")
MASK OFF ID
A = GRN
GRN vs logr
<
≥
;6
GRN vs higr
≥
<
IS GRN WITHIN THE GROUP # LIMITS THAT PERTAIN TO SCAN TYPE ESTY?
NO
YES
;7
COMPUTE IN B-REG ADDR = EL2 + 11·(GRN-1)
BY CALLING E2ADR
CANCEL
nleft ← nleft - 1
nleft = φ ?
YES
NO
TALLY
3
FETCH MXSN AND MNSN TO (A, E)
DOUBLE WORD LOAD OF MAX & MIN SCAN PERIOD FROM EL2(GRN)
;8
spud vs MXSN
>
≤
;9
spud vs MNSN
<
≥
IS SCAN PERIOD ESPD (spud) WITHIN LIMITS FOR GROUP # GRN?
NO
YES
4
KEEP
3

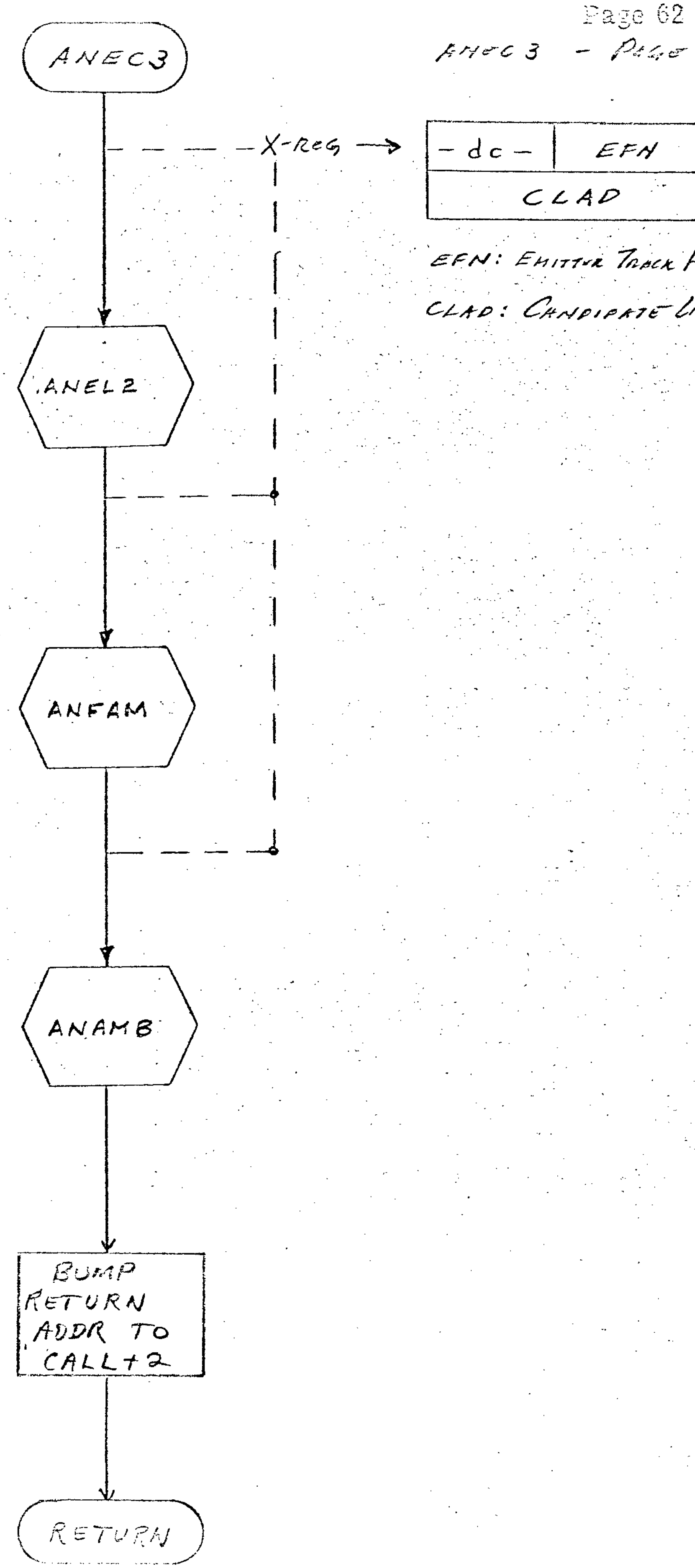

2
KEEP
STORE CAND. LIST
ITEM A (stpt)
stpt←stpt+1
X-REG COPY USED HERE
2
TALLY
ncand←ncand-1
TEST FOR DONE
WINNOW
2
NO
ncand = Φ
?
YES
TO NEXT
CAND. LIST
ITEM
;1Φ
STORE nleft
IN LEFT BYTE
OF (clad)
WINNOW LOOP
COMPLETE
;11
BUMP RETURN
ADDRESS TO
CALL+2
rtad ← rtad+1
4
ABNORM
CLEAN UP
STACK:
S-REG ←
S-REG + 8
S → clad
RESTORE
X, E, B, A-
REG'S FROM
STACK
S → rtad
RETURN


2
NOCAND
B-REG ← adef
= ETF + 16·EFN
STORE NOFA2 CODE IN ETF IDENT - EID
EZN ← Φ
"NONE-OF-ABOVE"
;12
STORE UNKNO CODE IN ETF DISPLAY CODE FIELD-EDIS
"UNKNOWN"
STORE Φ IN LEFT BYTE OF (c2ad)
ABNORM
RETURN TO CALL+1


E2APR
A-REG =
7
0
-dc-
INDEX
CLEAR
A-REG
LEFT BYTE
B ← A
B ← 11 * B
B ← B +
EL2 – 11
RETURN
A-REG =
INDEX
B-REG =
EL2 + 11(INDEX – 1)

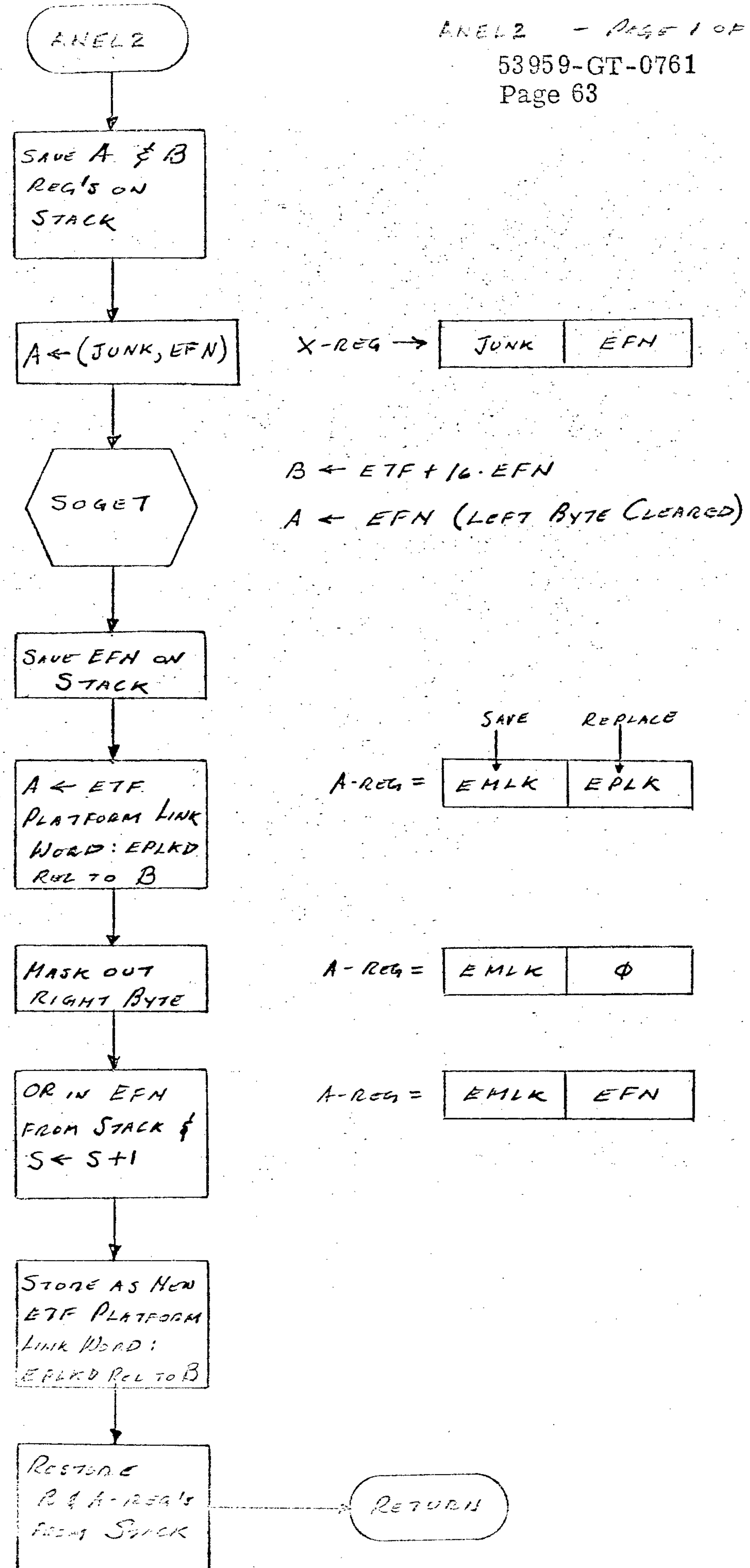

ANEL11
DUMMY
RETURN

3.2.10 ANEC3



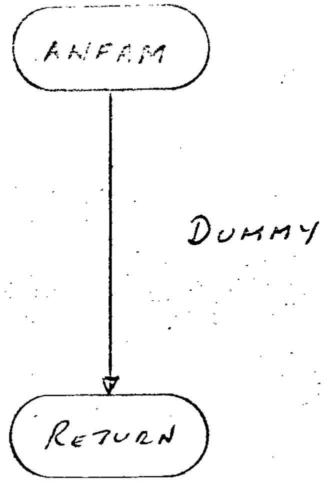
EFN: EMITTER TRACK FILE #
CLAD: CANDIDATE LIST ADDR.

SJW 10/14/76
REV TLC 11/15/76

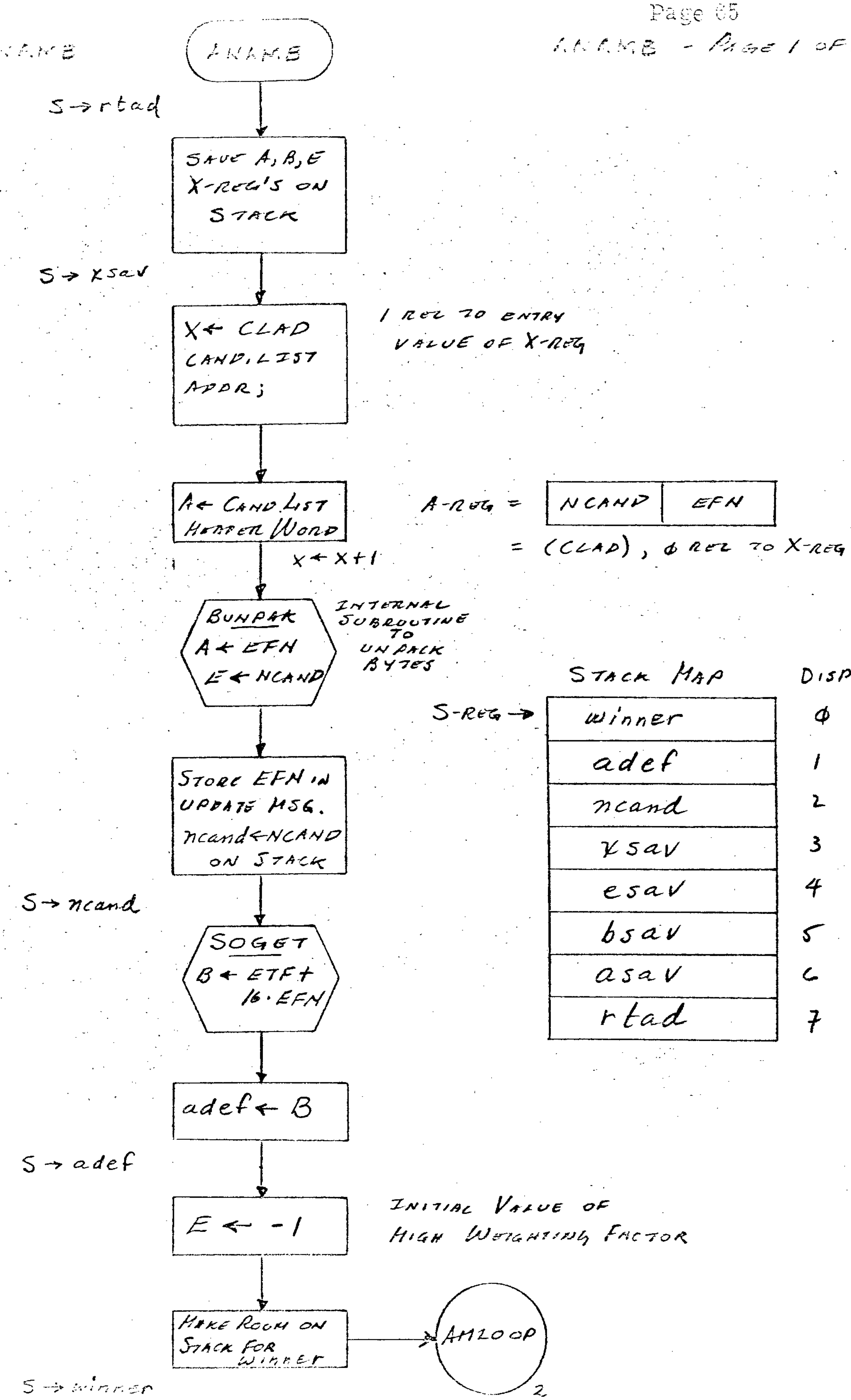

ANEL2
SAVE A & B REG'S ON STACK
A ← (JUNK, EFN)
X-REG →
JUNK
EFN
SOGET
B ← ETF + 16·EFN
A ← EFN (LEFT BYTE CLEARED)
SAVE EFN ON STACK
A ← ETF PLATFORM LINK WORD: EPLKD REL TO B
SAVE
REPLACE
A-REG =
EMLK
EPLK
MASK OUT RIGHT BYTE
A-REG =
EMLK
Φ
OR IN EFN FROM STACK & S ← S+1
A-REG =
EMLK
EFN
STORE AS NEW ETF PLATFORM LINK WORD: EPLKD REL TO B
RESTORE B & A-REG'S FROM STACK
RETURN

SSW 10/14/76

3.2.10.2 ANFAM



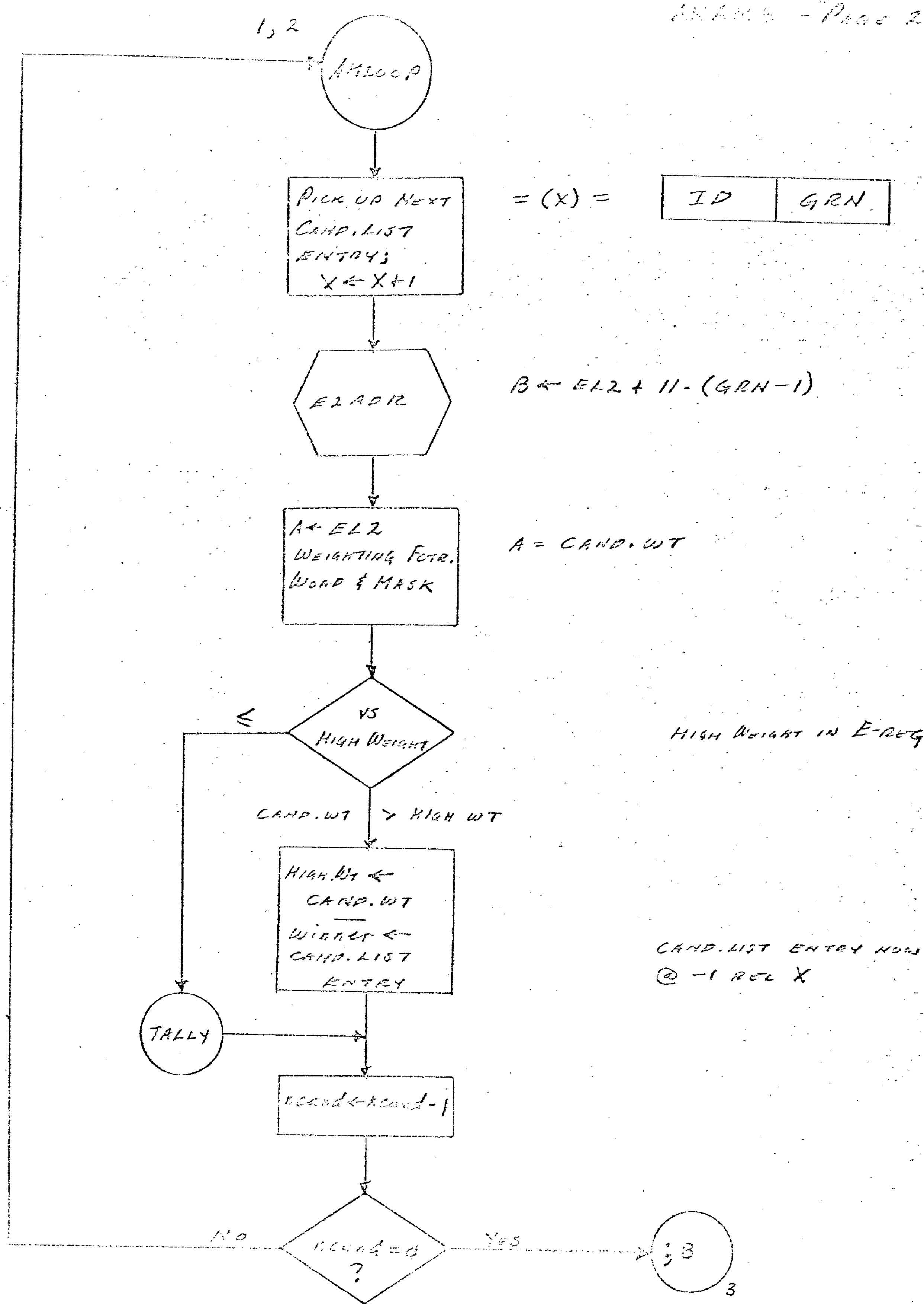
3.2.10.3 ANAME

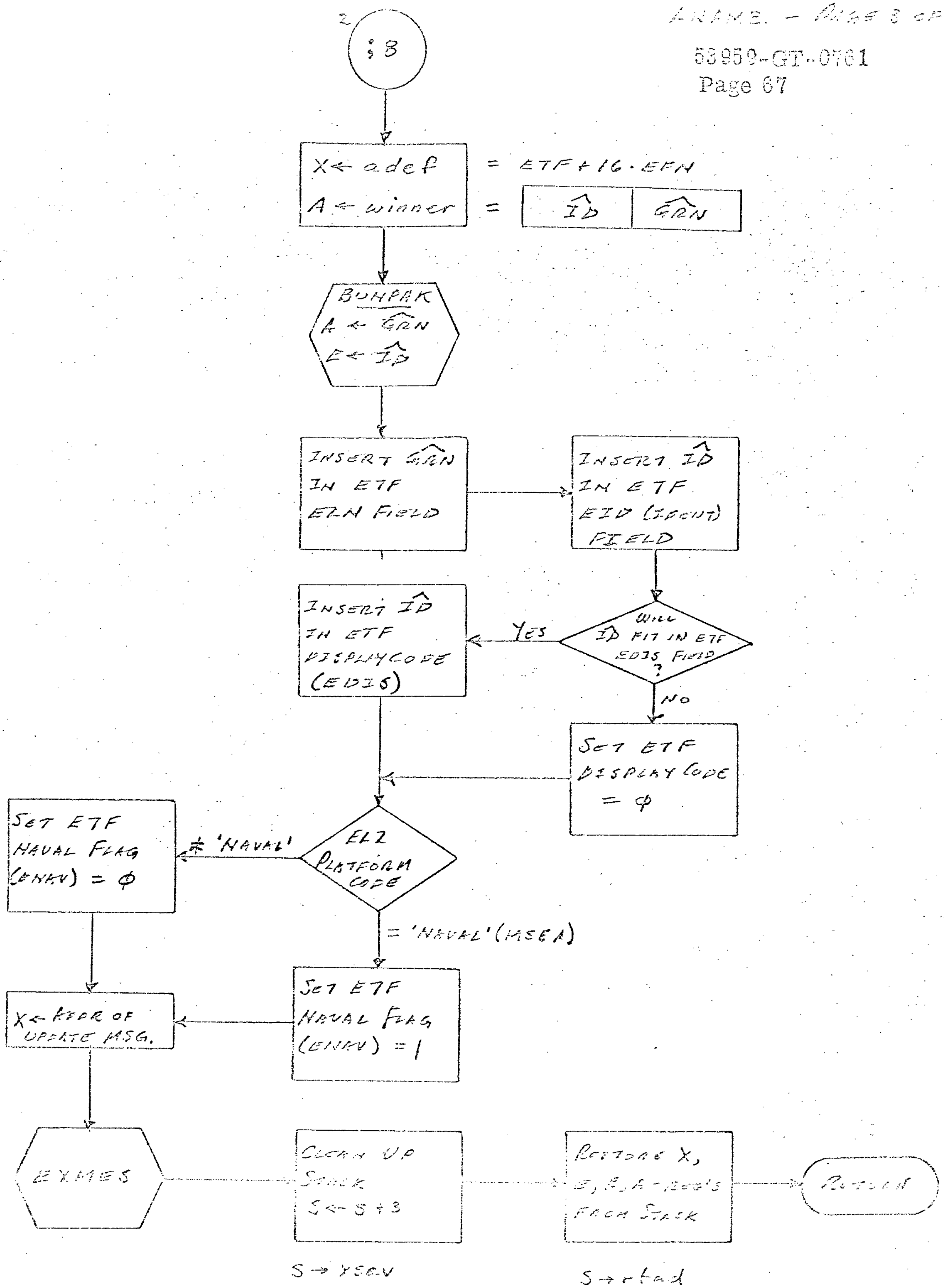


STACK MAP

	STACK MAP	DISP
S-REG →	winner	Φ
	adef	1
	ncand	2
	xsav	3
	esav	4
	bsav	5
	asav	6
	rtad	7

EJW 10/14/76


1, 2
AKLOOP
PICK UP NEXT CAND. LIST ENTRY; X ← X+1
= (X) =
ID
GRN.
E2ADR
B ← EL2 + 11 - (GRN-1)
A ← EL2 WEIGHTING FCTR. WORD & MASK
A = CAND. WT
VS HIGH WEIGHT
HIGH WEIGHT IN E-REG
≤
CAND. WT > HIGH WT
HIGH WT ← CAND. WT
WINNER ← CAND. LIST ENTRY
CAND. LIST ENTRY NOW @ -1 REL X
TALLY
NCAND ← NCAND - 1
NO
NCAND = 0 ?
YES
; B
3


2
;8
X ← adef
A ← winner
= ETF + 16 · EFN
= ID GRN
BUNPAK
A ← GRN
E ← ID
INSERT GRN IN ETF ERN FIELD
INSERT ID IN ETF EID (IDENT) FIELD
WILL ID FIT IN ETF EDIS FIELD ?
YES
NO
INSERT ID IN ETF DISPLAYCODE (EDIS)
SET ETF DISPLAY CODE = φ
ELZ PLATFORM CODE
≠ 'NAVAL'
= 'NAVAL' (MSEA)
SET ETF NAVAL FLAG (ENAV) = φ
SET ETF NAVAL FLAG (ENAV) = 1
X ← ADDR OF UPDATE MSG.
EXMES
CLEAN UP STACK S ← S + 3
RESTORE X, E, A - REG'S FROM STACK
RETURN
S → YSAV
S → rtad

3.3 COMPUTER SUBPROGRAM ENVIRONMENT

3.3.1 Tables

3.3.1.1 Analysis Return Driver Table

Analysis Return Processing Table (ANMPT)

Purpose and Type -

Fixed length table containing the addresses of the subroutines called to process an Analysis Return message.

Size and Indexing Procedure -

Nine entries of one 16-Bit word. All entries shall be referenced by indexed displacement from the start of the table.

Entry Format -

15 — 0
Routine Addr

Field	Description	Units	LSB
Routine Addr	Address of an analysis return message processing routine	N/A	N/A

3.3.1.2 NE Processing 2 Tables

None.

3.3.1.3 NE Processing 3 Tables

None.

3.3.1.4 NOFA2 Processing 2 Tables

None.

3.3.1.5 NOFA2 Processing 3 Tables

None.

3.3.1.6 EOC Processing 2 Tables

None.

3.3.1.7 EOC Processing 3 Tables

None.

3.3.1.8 EOC Processing 4 Tables

None.

3.3.1.9 Emitter Classification 2 Tables

3.3.1.9.1 Contemporaneous Analysis Request Message

a) The name of this table is CRQMSG. It is local to ANEC2.

b) CRQMSG is used to hold two fixed constants and three variable words filled in and by ANEC2, the totality constituting a message to the Executive stating that contemp. analysis is or is not wanted. The location of the message is made known to ANEC2's caller by returning the address (CRQMSG) of its 1st word in the X-Reg.

c) CRQMSG is of fixed length = 5. It is indexed by use of individual labels attached to the locations requiring access.

d) CRQMSG's structure and Bit layout is shown in the accompanying diagram.

LABEL	CONTENTS		EXPLANATION	
CRQMSG:	EMNCAQ		EXECUTIVE MESSAGE # FOR CONTEMP. ANALYSIS REQUEST.	
	3		# OF WORDS TO FOLLOW	
CRRMCD:	15 RMCEC2 8	7 EFN 0	RETURN MODULE CODE FOR E.C. 2	EMITTER TRACK FILE NUMBER
CRCLAD:	CLAD		CANDIDATE LIST ADDRESS	
CRREQW:	AN (bit 15)	1 (bit 3)	AW: =1 ANALYSIS WANTED =0 ANALYSIS NOT WANTED BIT 3: ANALYSIS TYPE IS CONTEMP.	

3.3.1.9.1 CONTEMPORANEOUS ANALYSIS REQUEST MESSAGE

3.3.1.9.2 Scan Type Outer Directory to EL2

a) The name of this table shall be ODA. ST

b) The purpose of ODA. ST is based on the requirement that the 11-word files of EL2 be sorted on ascending scan type. Then each element of ODA. ST, say the I-th, I = Ø, 1, ... 15 contains the lowest file number in EL2 that exhibits scan type = I. The 17th entry (I = 16) of ODA. ST contains the number N + 1 where N = the number of files of EL2.

Thus, in ANLV2 to eliminate candidates on the basis of exact match to the current scan type of the subject ETF file (ESTY) we look up.

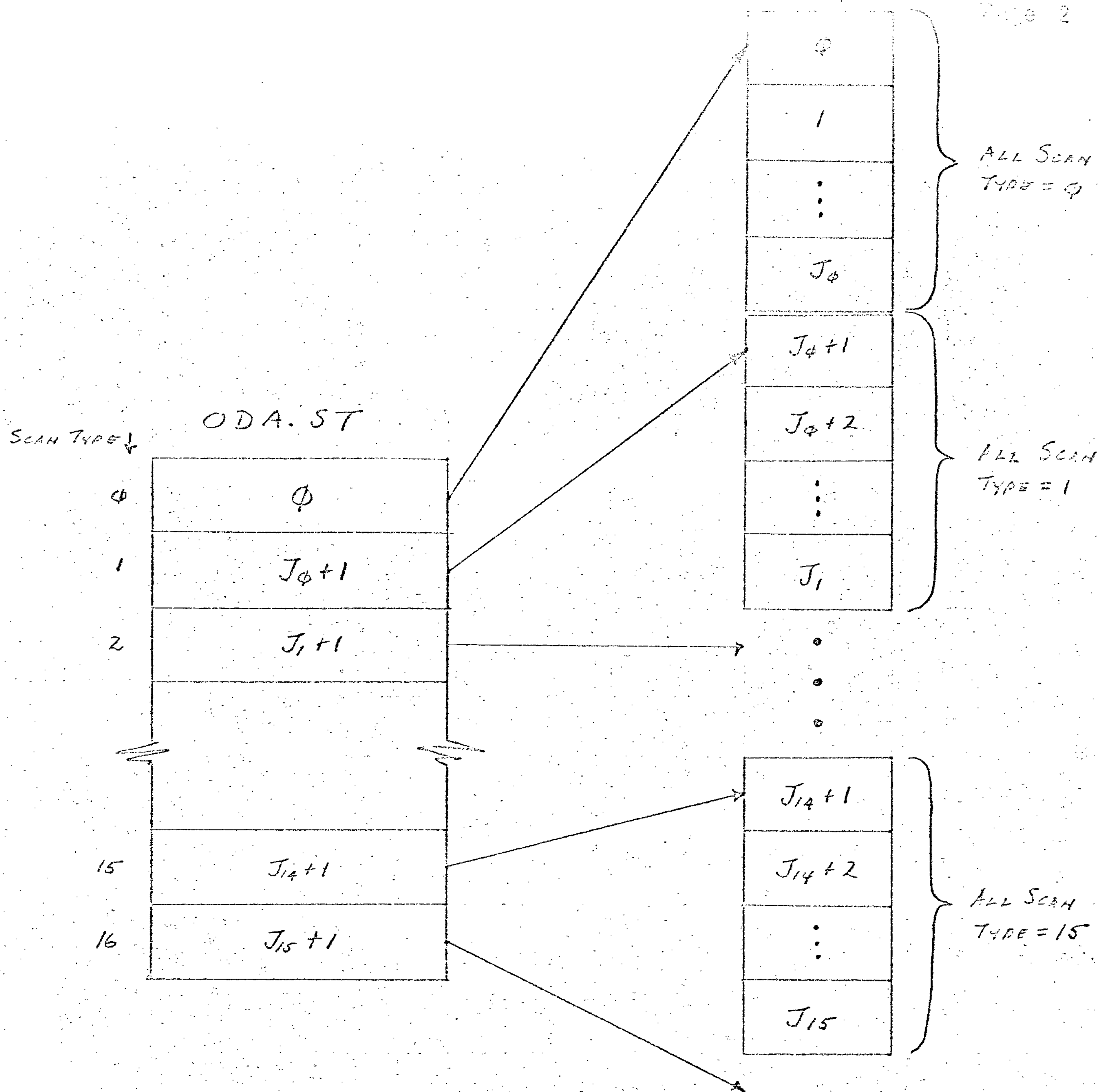
logr = (ODA. ST + ESTY) and

higr = (ODA. ST + ESTY + 1)

and ask if the candidate group # is such that logr ≤ group # < higr (yes-keep; no-cancel)

c) ODA. ST shall be of length = 17 based on allocation of 4 bits to scan type. It shall be indexed by anding an index I = Ø, 1, ... 16, to address ODA. ST.

d) The structure of ODA. ST and its relation to EL2 are shown in the accompanying diagram. Bit layout is not applicable since each element is a whole word item.



3.3.1.8.2 ODA.ST STRUCTURE & RELATION TO EL2

3.3.1.10 Emitter Classification 3 Tables

3.3.1.10.1 Update Message -

a) The name of this table shall be UPMSG. It shall be local to subroutine ANAMB.

b) The purpose of UPMSG shall be to inform the executive that classification has been completed on the emitter whose track file # was input to ANAMB, so that the Executive may take and/or schedule those actions which properly emanate from said event.

c) UPMSG shall be three words long and shall be indexed by use of labels attached to those entries which require access.

d) Structure and Bit layout shall be as shown:

Label	Contents	
UPMSG:	EMNEC3	Executive Message #
	1	# of words to follow
UPEFN:	EFN	Stored by ANAMB

3.3.2 Variables

3.3.2.1 Analysis Return Driver Variables

None.

3.3.2.2 New Emitter Processing 2 Variables

ANNE2 variable are defined in Table I.

3.3.2.3 New Emitter Processing 3 Variables

ANNE3 variables are defined in Table II.

TABLE I

VARIABLE DESCRIPTIONS FOR NEPROC2

Descriptive Item	Variable Name		
	EFN	EFP	
Purpose	Value of ETF entry	Provides address of first word of ETF entry given by EFN	
Type	Fixed point	Fixed point	
Size	8	16	
Binary Pt.	Bit 0	Bit 0	
Max. Value	127	65,536*	
Min. Value	-128	0*	
Initial Value	Don't care	Don't care	
Static/ Dynamic	Dynamic	Dynamic	

* Memory map assignment will restrict this.

TABLE II

VARIABLE DESCRIPTIONS FOR NE PROC 3

Descriptive Item	Variable Name					
	EFN	EFP	GDQ	M	PARAM	QUAL
Purpose	Value of ETF entry	Provides address of 1st word of ETF entry given by EFN	Indicator of data quality	One less than number of significant bits in PARAM.	Parameter which is to be tested for quality	Quality factor associated with PARAM.
Type	Fixed point	Fixed point	Fixed point	Fixed point	Fixed point	Fixed point
Size	8	16	1	4	16	4
Binary Point	Bit 0	Bit 0	N/A	Bit 0	Bit 0	Bit 0
Max. Value	127	65,536*	1=good quality	15	65,536	15
Min. Value	-128	0*	0=bad quality	3	0	0
Initial Value	Don't care	Don't care	Don't care	Don't care	Don't care	Don't care
Static/ Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic

* Memory map assignment will restrict this.

CLEARPRINT VELLUM PRINTED IN U.S.A.

3.3.2.4 NOFA Process 2 Variables

1) NOFA2 Process 2 Contemporaneous Analysis Request Message (ANNCA)

ANNA2 may generate a contemporaneous analysis request message (which will then be sent to the EXEC by the AR driver). This message has the format of an Analysis Request Message (see Figure 2), with

ANNW = 3

ANRMC = X'Ø4'

ANCA = 1

ANAW = Ø or 1

2) △SPRD

ANDSP is used by ANNA2 to determine if the scan analysis scan period measurement differs significantly from the scan period stored in the emitter track file.

15	Ø
ANDSP	

Field	Description	Units	LSB
ANDSP	Delta SPRD	Msec	1/4

3.3.2.5 NOFA2 Process 3 Variables

None.

3.3.2.6 EOC Process 2 Variables

3.3.2.6.1 Update Message (ANUPM) - ANOC2 may generate an update message and send it to the EXEC. Format is shown in Figure 4.

3.3.2.6.2 Contemporaneous Analysis Request Message (ANOCA) - ANOC2 may generate a contemporaneous analysis request message. This message has the format of an Analysis Request Message (see Figure 2) with:

ANNW = 3
ANRMC = X'Ø6' or X'Ø7'
ANCA = 1
ANAW = Ø or 1

3.3.2.7 EOC Process 3 Variables

None.

3.3.2.8 EOC Process 4 Update Message (ANUPM)

ANOC4 may generate an update message and send it to the EXEC. Format is shown in Figure 4.

3.3.2.9 Emitter Classification 2 Variables

Only subroutine ANLV2 has any local variables, i.e., entities stored and retrieved from memory. These variables are all maintained on the stack during ANLV2's execution and their space is relinquished before exiting.

A stack map is shown in the accompanying figure. It, and the text to follow employ the following convention:

A symbolic displacement (for use in S-indexed access instructions) is shown in upper-case. The contents of such a location are denoted by the same symbol written in lower-case.

3.3.2.9.1 Variables of Subroutine ANLV2 - The following items are described in top-of-stack to bottom order.

logr - Low Group Number = (ODA. ST + ESTY) = smallest group # that can exhibit scan type = ESTY.

higr - High Group Number = (ODA. ST + ESTY + 1) = one more than largest group # that can exhibit scan type = ESTY.

spud - Temporary repository for ETF scan period (ESPD).

adef - Temporary repository for ETF (EFN) address. Needed only if all candidates are eliminate and failure codes must be stored back in ETF (EFN).

nleft - This variable and the next are both originally set = the number of candidates in the input candidate list (left byte of 1st word thereof).

nleft is decremented by 1 each time a candidate is eliminated and immediately thereafter the new value of nleft is tested for equality to Ø. If equality is attained, the candidate list has been entirely eliminated and the algorithm ends: See steps NOCAND - 13 in 3. 1. 9. 2.

If the algorithm ends with nleft > Ø, then nleft is size of the reduced (or possibly same size) candidate list. This value is stored in the left byte of the word whence nleft was initialized.

ncand - Is initialized as described under nleft.

ncand is decremented by 1 after the consideration of each candidate whether kept or cancelled. When ncand goes to Ø, the algorithm ends, unless ended prematurely as described under nleft.

stpt - This variable and the next, are both originally set = ADDR of 1st Cand. List entry (2nd word of list, immediately after header word).

continued-	-	stpt is incremented by 1 after each store of a kept (retained) candidate back into the list. Note that at all times clad + 1 ≤ stpt ≤ rdpt, where clad and rdpt are described below.

rdpt - Is initialized as described under stpt.

rdpt is incremented by 1 for each candidate list entry fetched for consideration. This is done after rdpt has been used as an indirect address to fetch the Cand. List entry = (rdpt).

clad - Pointer to the header word of the input Cand. List as received upon entry in the X-Reg.

$\begin{Bmatrix} \text{esav} \\ \text{bsav} \\ \text{asav} \end{Bmatrix}$ - Save input contents of $\begin{Bmatrix} \text{E-Reg} \\ \text{B-Reg} \\ \text{A-Reg} \end{Bmatrix}$

Not referenced by name.

rtad - Return address. Accessed by name on a normal return to call +2:

rtad ← rtad +1

3.3.2.1 ECLV2 Stack Map

	Displacements Symbolic	Displacements Numeric
S-REG → logr	LOGR	0
higr	HIGR	1
spud	SPUD	2
adef	ADEF	3
nleft	NLEFT	4
ncand	NCAND	5
stpt	STPT	6
rdpt	RDPT	7
clad	CLAD	8
esav	ESAV	9
bsav	BSAV	10
asav	ASAV	11
rtad	RTAD	12

3.3.2.10 Emitter Classification 3 Variables

No permanent space shall be allocated to non-tabular data. Three temporary local variables shall be used by subroutine ANAMB. Space for them shall be allocated on the stack during initialization and relinquished prior to Exit. The stack map shall be as shown on the first page of the ANAMB flow chart, 3.3.2.10.3.

The variables shown there are:

Winner - During execution of the loop which searches for the maximum weighting factor over the input set of candidates, winner shall be set = the Candidate List entry word of the each candidate whose weighting factor exceeds the maximum factor found up to that point. Note that the maximum weight shall be initialized = -1 guaranteeing that the first Candidate List entry, at least, will be stored at winner.

adef - Shall be used to hold for later use the emitter track file base address for the EFN-TM file:

$$ETF\ (EFN) = ETF + 16 \cdot EFN$$

ncand - Shall be initialized with the Candidate List length as extracted from the left byte of the header word thereof.

Thereafter, ncand shall be used as a loop iteration control; ncand ← ncand ← -1 and repeat loop if ncand $\neq 0$.

3.3.3 Constants

There are no local constants associated with the Analysis Return Functional Group.

3.3.4 Flags

There are no local flags associated with the Analysis Return Functional Group.

3.3.5 Indices

The Emitter File Number (EFN), is an idex that is used throughout the Analysis Return Functional Group. It is used to access an entry in the Emitter Track File (EF). EFN assumes the following range of values: $\emptyset \leq EFN \leq 127$.

3.3.5.1 Analysis Return Driver Indices

Analysis Return Message Processing Table Index:

a) Index Name. I (Not a symbolic label)

b) Purpose. This index is used to fetch an Analysis Return message processing routine address from table ANMPT. "I" assumes the following range of values:

$$1 \leq I \leq 9$$

3.3.6 Common Data Base References

3.3.6.1 Analysis Return Driver (ANDR) Common Data Base References

None.

3.3.6.2 New Emitter Processing 2 (ANNE2) Common Data Base References

1) Emitter Track File (EF)

3.3.6.3 New Emitter Processing 3 (ANNE3) Common Data Base References

1) Emitter Track File (EF)

3.3.6.4 NOFA2 Process 2 (ANNA2) Common Data Base References

1) Emitter Track File (EF)

3.3.6.5 NOFA2 Process 2 (ANNA3) Common Data Base References

None.

3.3.6.6 EOC Process 2 (ANOC2) Common Data Base References

1) Candidate List (CL)

2) Emitter Track File (EF)

3.3.6.7 EOC Process 3 (ANOC3) Common Data Base References

None.

3.3.6.8 EOC Process 4 (ANOC4) Common Data Base References

None.

3.3.6.8.1 Update Link Analysis 2 Common Data Base References

1) Emitter Track File (EF)

3.3.6.9 Emitter Classification 2 Common Data Base References

SUBROUTINE

Item		ANEC2	ANST2	ANLV2	ANEL1
ETF	EDIS			S	
	EID			S	
	ESPD		S	U	
	ESTY		S/U	U	
EL2	MXSN			U	
	MNSN			U	
Parameters					
	ESDLB		✓		
	NUL		✓		
	ENA2			✓	
	EUNK			✓	
✓, U: Used S: Set					

3.3.6.10 Emitter Classification 3 Common Data Base References

SUBROUTINE

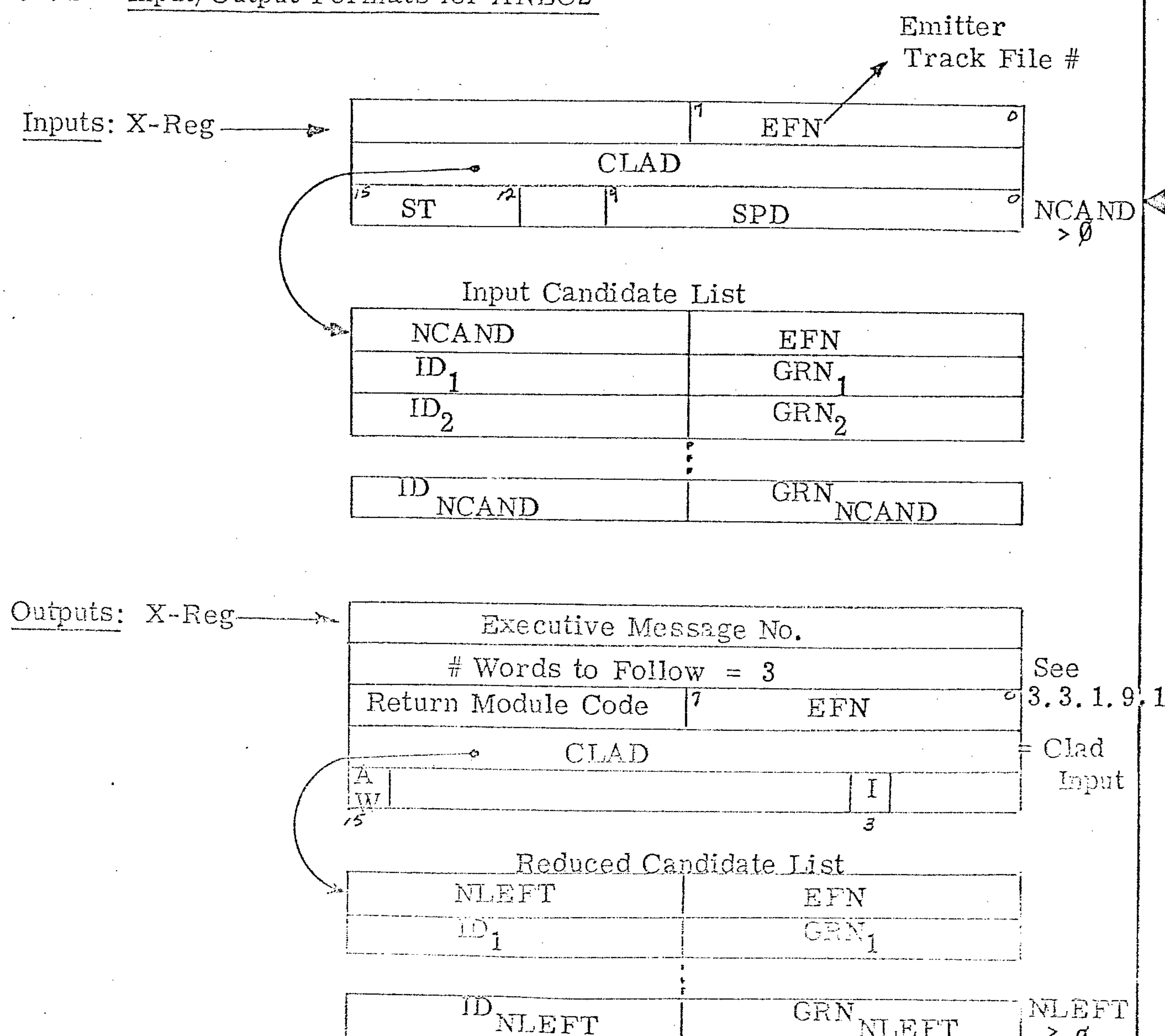
Item		ANEC3	ANEL2	ANFAM	ANAMB
ETF	EPLK		S		
	EDIS				S
	EID				S
	ELN				S
	ENAV				S
EL2	MFCT				U
	MPLT				U
Parameters	NAVAL				✓

3.4 INPUT/OUTPUT

The format of all input and output messages shall be as specified below:

Item	Input or Output	Specification Document
Analysis Return Message	Input	CDBDD, 53959-GT-0751
Analysis Request Message	Output	"
Classification Message	Output	"
Instrumentation Data	Output	Data Extraction CSDD, 53959-GT-0759

3.4.1 Input/Output Formats for ANEC2



3.4.1.1 ANST2 Input/Output

Inputs: X-Reg exactly as input to ANEC2.

Outputs: Possible changes to

- ETF Scan Type & Period
- ETF Scan Type & Period (in SCTCOM if called)
- ETF State Indicator

3.4.1.2 ANLV2 Input/Output

Inputs: X-Reg → Candidate List as illustrated in inputs for ANEC2.

Outputs: X-Reg (Unchanged) points to reduced Candidate List illustrated in Section 3.4.1.

3.5 REQUIRED EXTERNAL SUBROUTINES

3.5.1 SOGET

SOGET is a subroutine in Sorter Message Processing (Document No. 53959-GT-0755) called by ANST2, ANLV2, ANNA2, ANEC2, and others.

Input: EFN whole word item in A-Reg.

Output: ETF + 16 * EFN in B-Reg.

3.5.2 SCTCOM

SCTCOM (Scan Test Common Logic) is a subroutine in Emitter Classification Processing -1 (Document No. 53959-GT-0760) which, as implied by its title, is shared with ECST1 (Scan Test 1).

SCTCOM requires the address ETF + 16 * EFN to be in X-Reg on entry. It complements the ETF State Indicator (ESIN) and if now on (= 1), sets the ETF Scan Type (ESTY) to circular and the ETF Scan Period (ESPD) to Time-Out.

3.6 CONDITIONS FOR INITIALIZATION

This subprogram shall have unconditional entry and shall require no special initialization procedure.

3.7 SUBPROGRAM LIMITATIONS

The Analysis Return Functional Group shall make the following assumptions and be subject to the following limitations:

1. ANDR retrieves the Return Module Code from the Analysis Return message and verifies that it is a valid code (= 1, 2, 3, . . . , 8, or 9). If not valid, an error alert message shall be sent to Instrumentation.

2. Emitter Classification 2 Algorithm Limitations - The algorithms in this subprogram are programmatic sequels to those of Emitter Classification Processing -1 (Document No. 53959-GT-0760), and are part of a single, overall search-and-classification strategy. Hence, the limitations on the algorithms stated in the referenced document carry over to here.

   The one local limitation that does stand out is that the length of ODA.ST is imposed by the allocation of 4-bits to scan type. If more than 16 types should be required in the future, ODA.ST would have to be lengthened.

   Note also, that the method of matching scan type using ODA.ST imposes a design requirement on EL2, namely

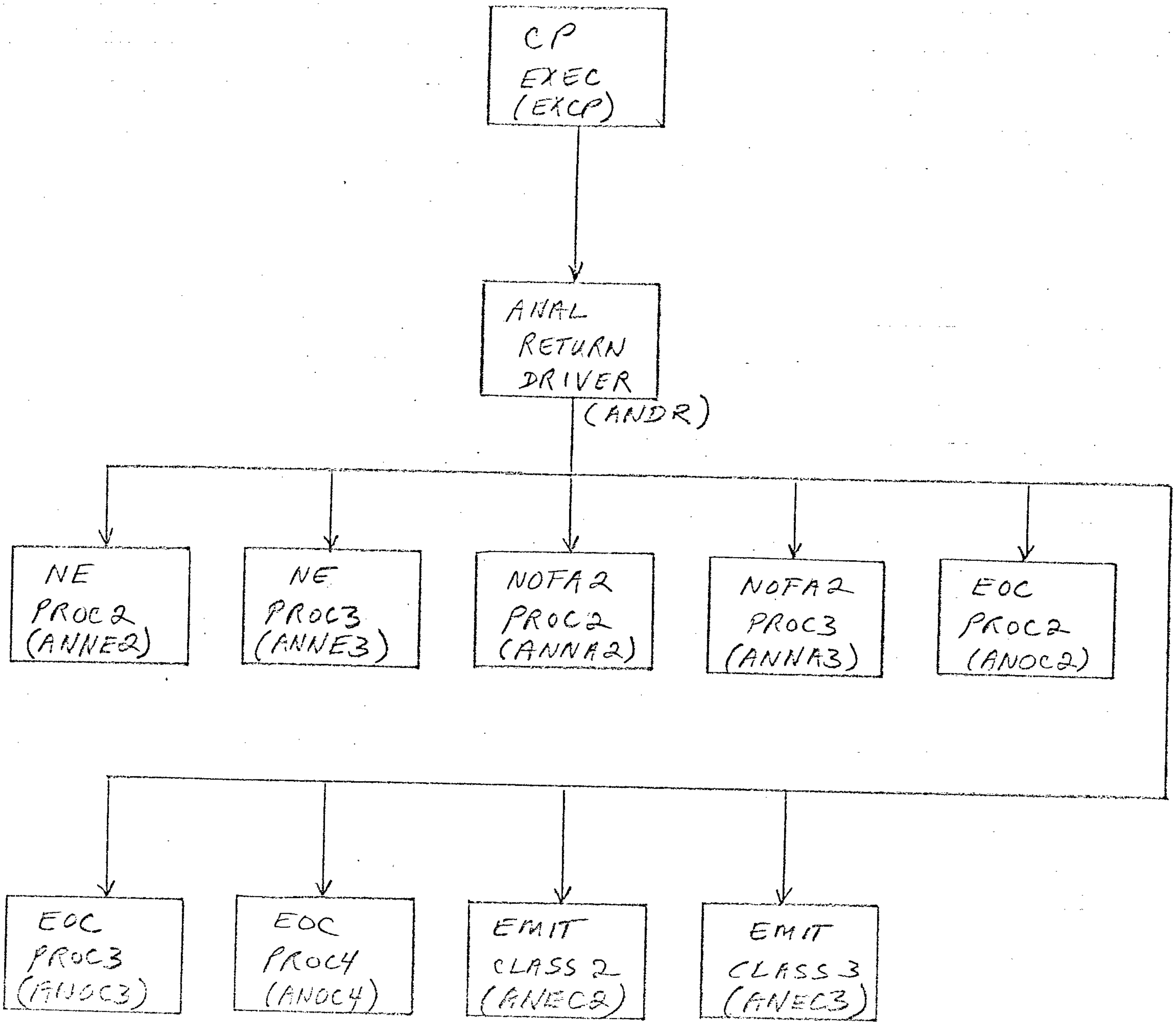
   EL2 files must be sorted on ascending scan type. (see Section 3.3.1.9.2).

PRINTED ON VELLUM PRINTED IN U.S.A.

3.8 INTERFACE DESCRIPTION

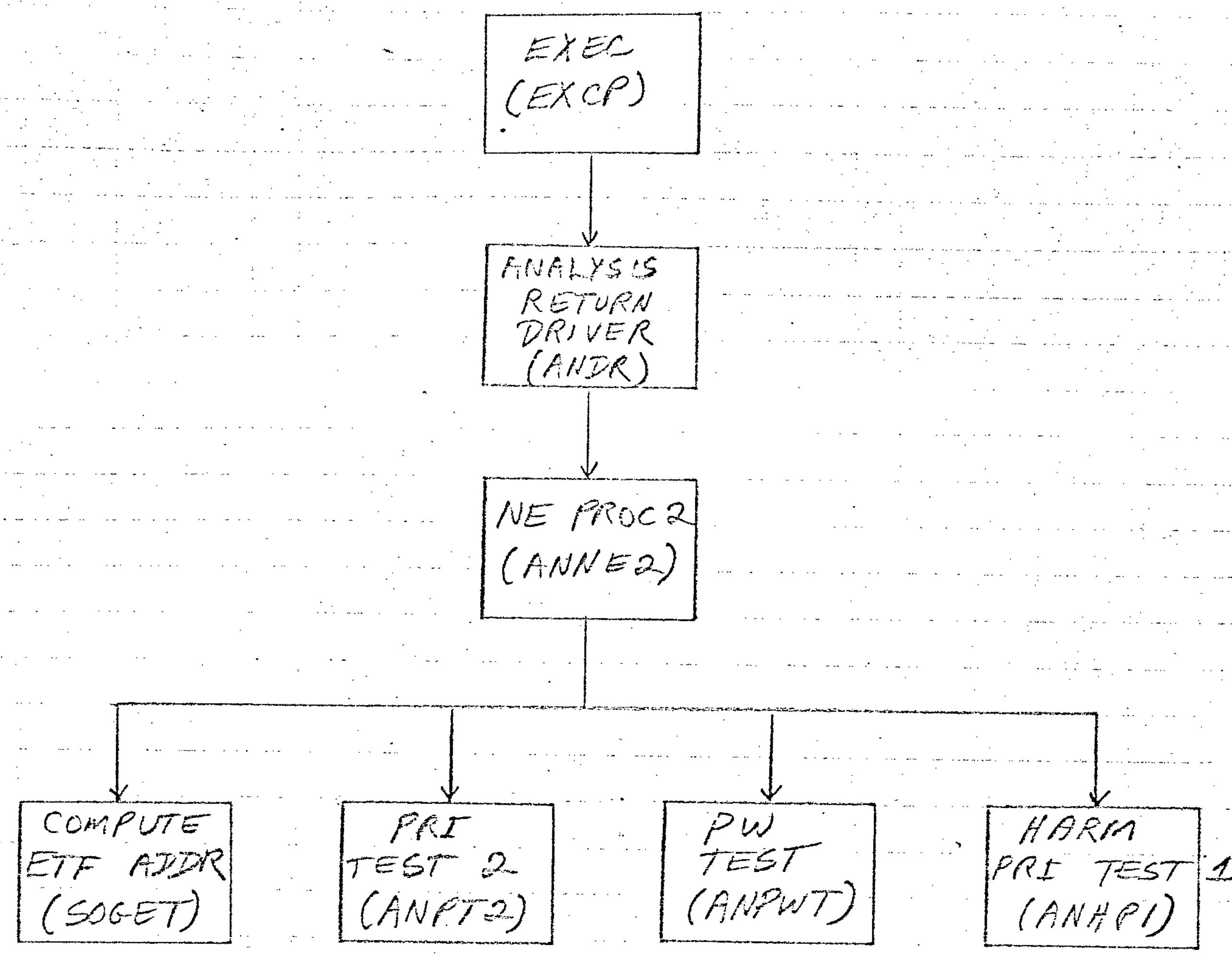
The Analysis Return Driver (ANDR) shall be called by the EXEC. ANDR shall then call one of the Analysis Return processing routines (ANNE2, ANNE3, ANNA2, etc.). The routines called by each Analysis Return processing routines are shown in the following interface diagrams. Instrumentation shall be called as required for data extraction and is not shown on the diagrams. Calls to the Executive message function are also not shown.

[illegible] PRINTED IN U.S.A.

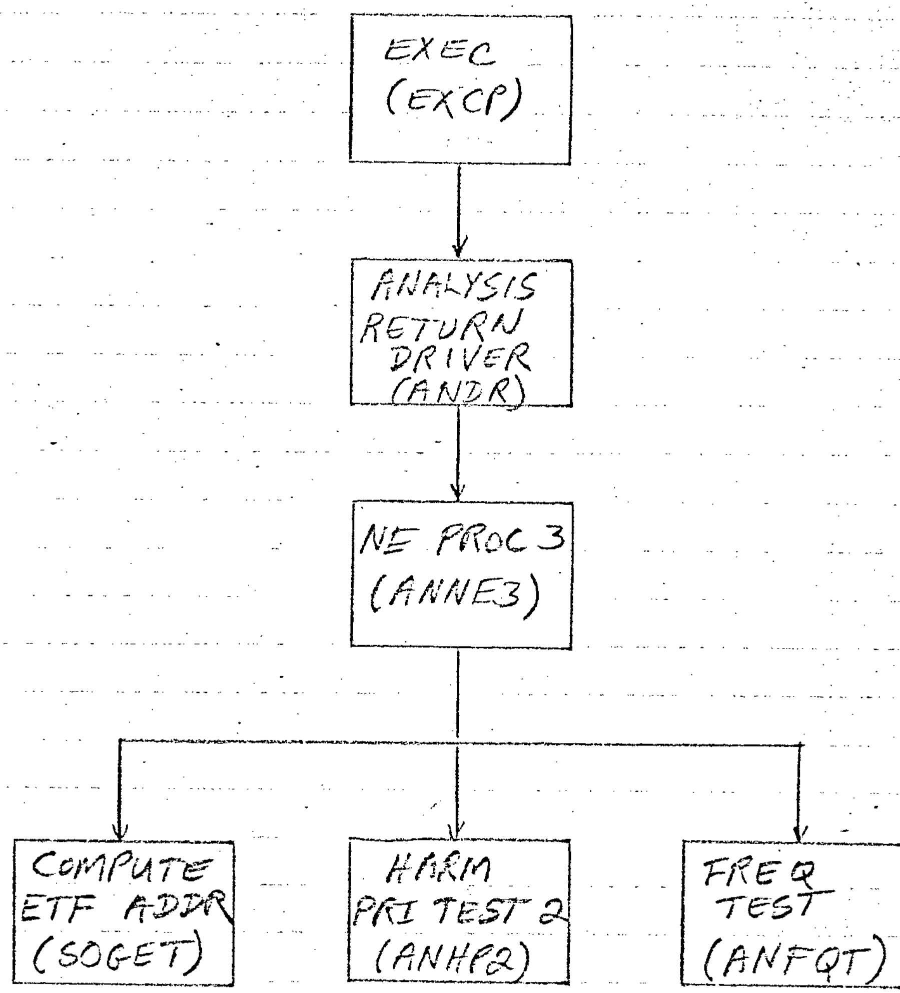


INTERFACE DESCRIPTION
ANALYSIS RETURN DRIVER

JLC 8 SEP 76

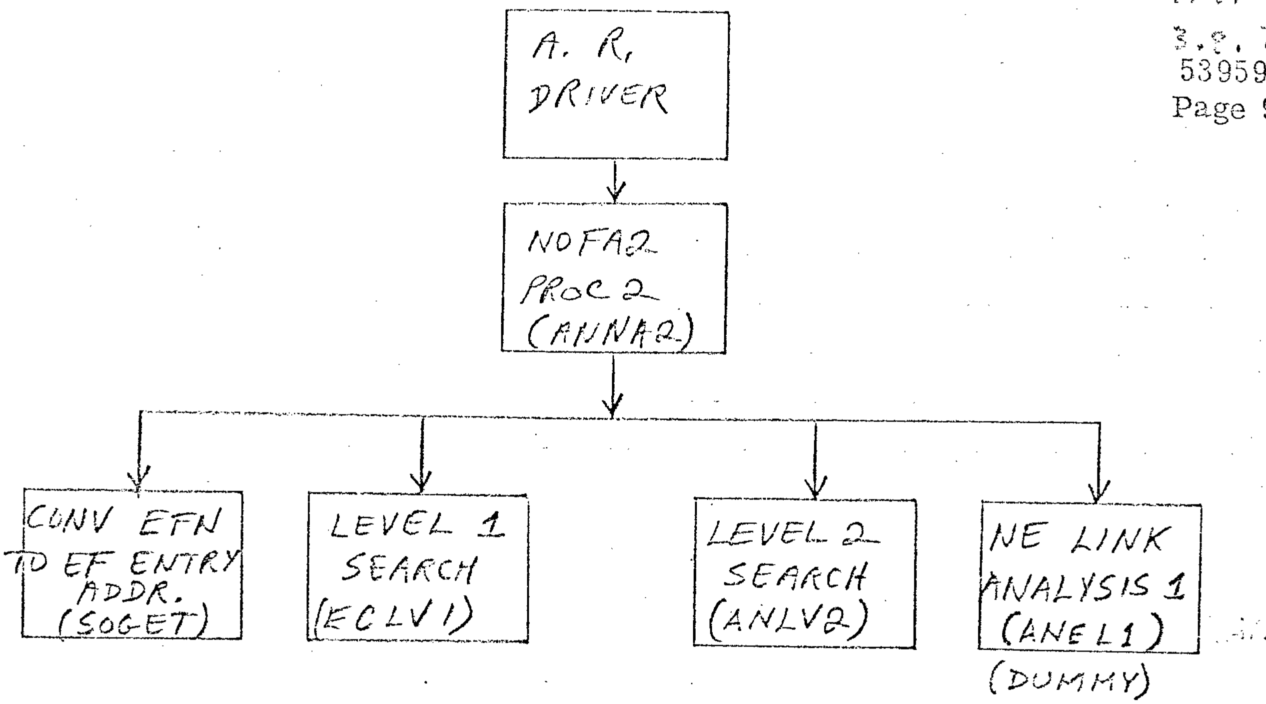


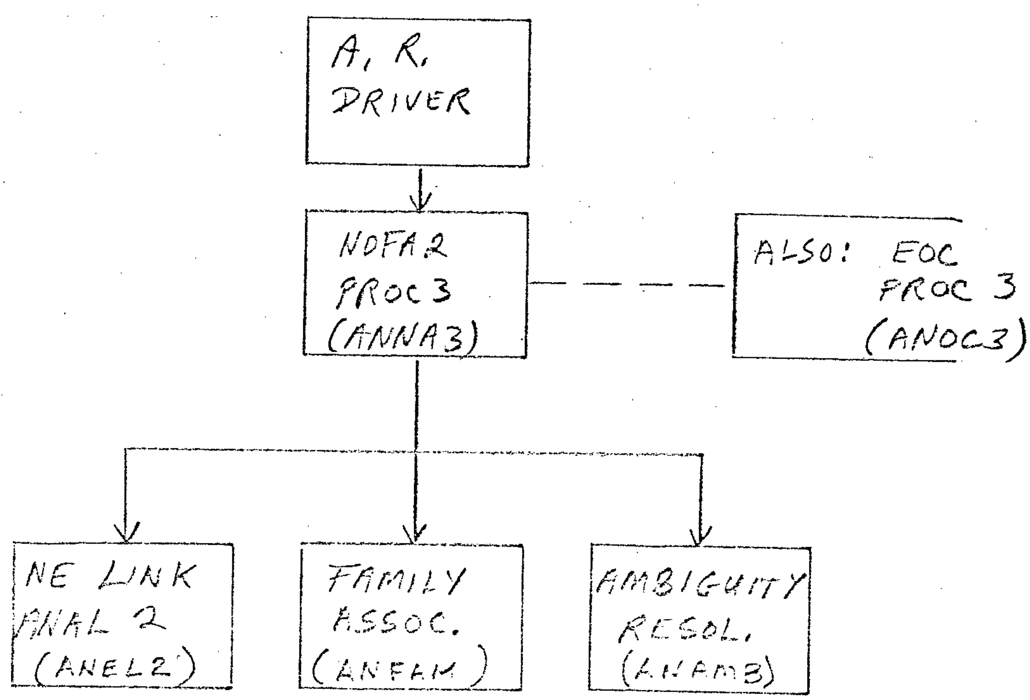
INTERFACE DESCRIPTION
NE PROCESSING 2
TLC 15 OCT 76



INTERFACE DESCRIPTION
NE PROCESSING 3

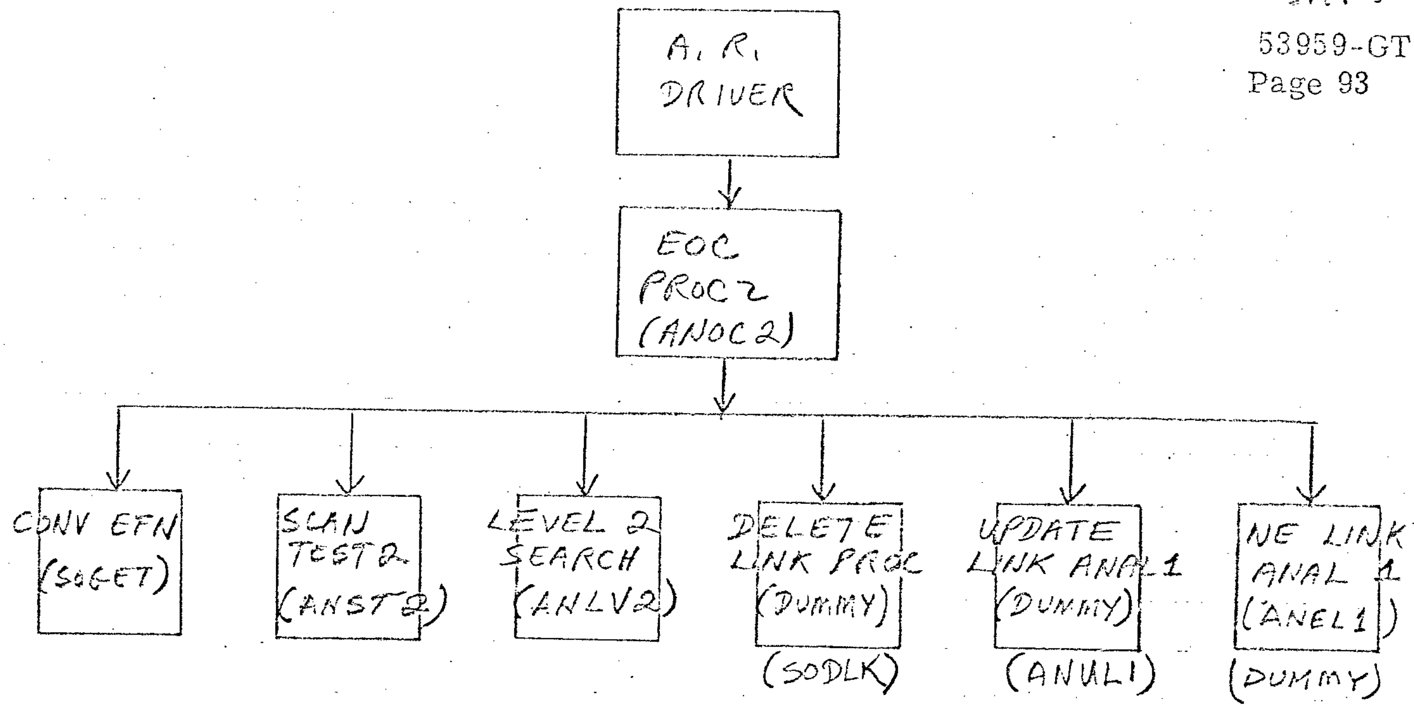
TLC 15 OCT 76

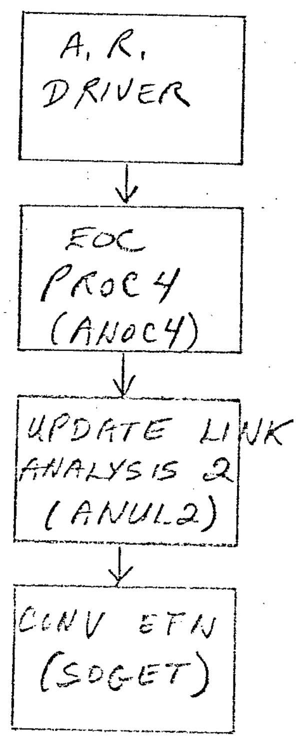




INTERFACE DESCRIPTION
- NOFA2 PROC2
- NOFA2 PROC3
- EOC PROC3

T.C. 8 SEP 76



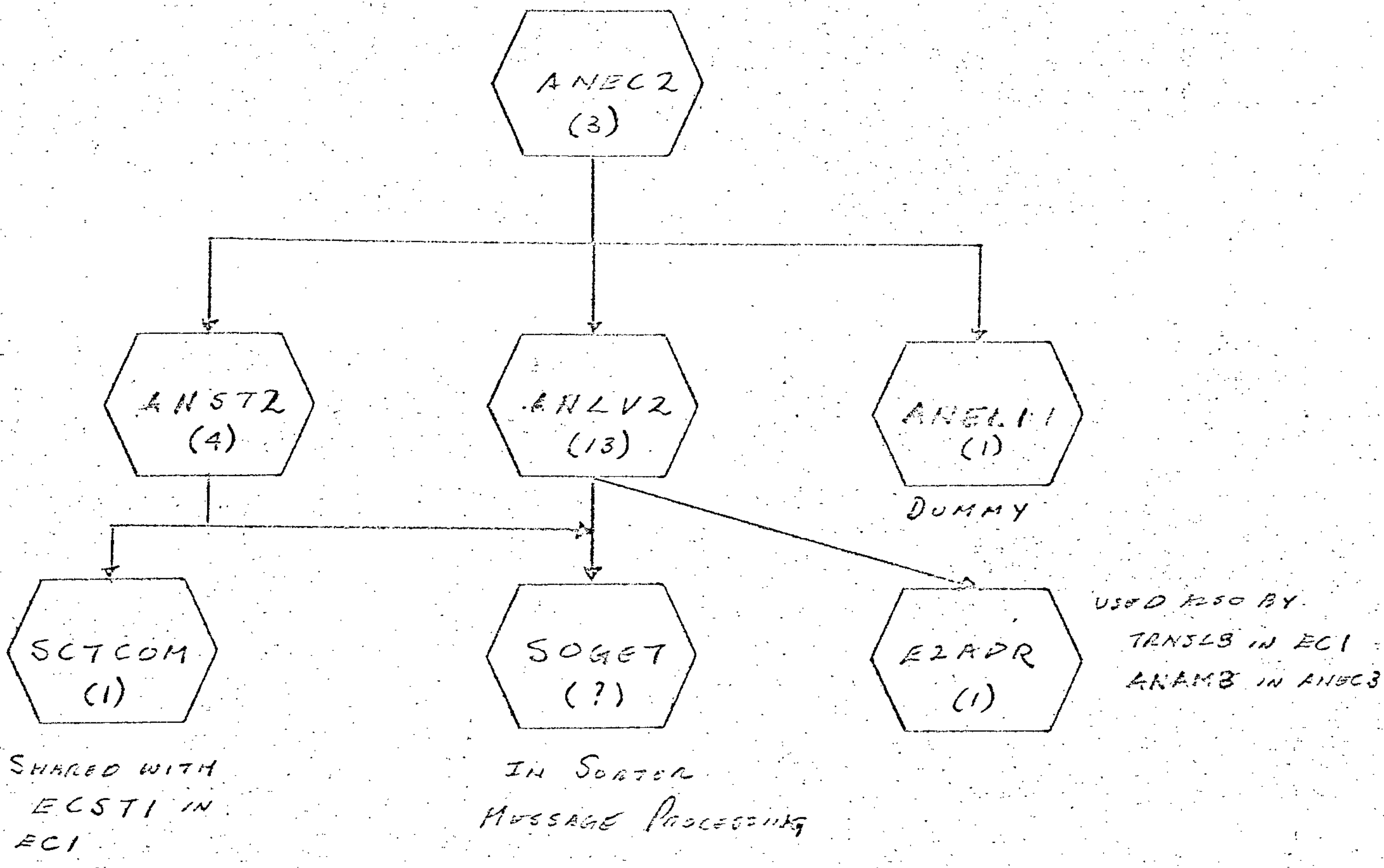


INTERFACE DESCRIPTION
— EOC PROC2
— EOC PROC4

TLC 8 SEP 76

EMITTER CLASSIFICATION 2

WHO CALLS WHOM / STACK DEPTHS



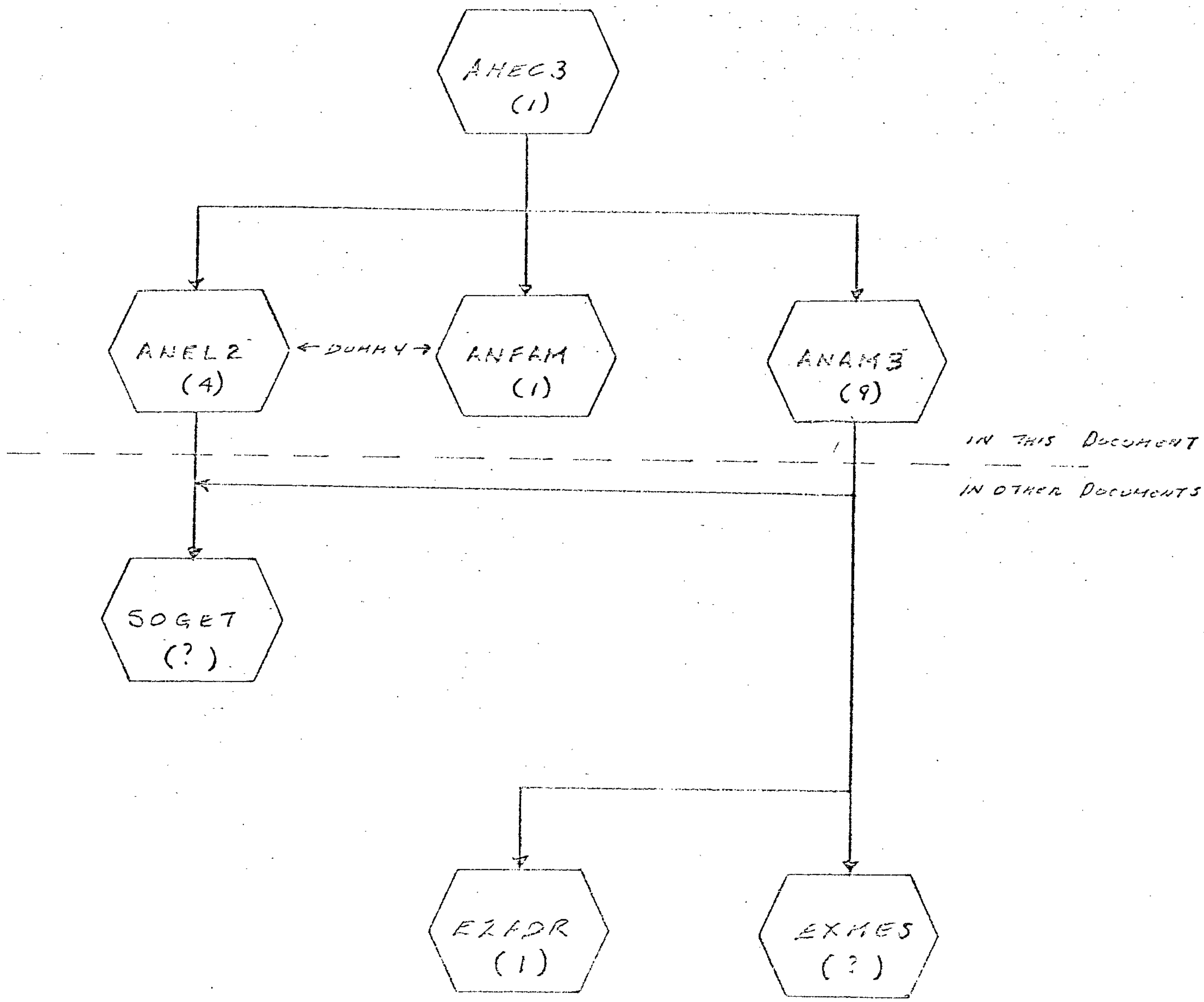
(N) = STACK DEPTH INCREASE (MAXIMUM) CAUSED BY CALL TO ROUTINE. THIS INCLUDES THE RETURN ADDRESS, BUT NOT ADDITIONS CAUSED BY FURTHER CALLS.

MAXIMUM STACK DEPTH $\geq$ 17 (AT LEAST 1 FOR CALL ON SOGET) ACHIEVED WHEN ANEC2 → ANLV2 → SOGET.

STRUCTURE DESCRIPTION
EMIT CLASS 2

EMITTER CLASSIFICATION PROCESSING - 3

WHO CALLS WHOM & STACK DEPTHS



INTERFACE DESCRIPTION

EMIT CLASS 3

SJW 10/14/76

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word Ø	ANMNO															
1	ANNW															
2	ANRMC								ANEFN							
3	ANPTR															
4	ANSTY				NOT USED		ANSPR									
5	NOT USED															
6																
7																
8																
9																
10																
11																
12																
13	NOT USED															

Figure 1a. (Scan) Analysis Return Message Format

Field	Description	Units	LSB
ANMNO	Executive Message No. (= 4)	N/A	1
ANNW	No. of Words in Message (= 3)	N/A	1
ANRMC	Return Module Code NEPROC2 = 1 EOC PROC3 = 6 NEPROC3 = 2 EOC PROC4 = 7 NOFA2 PROC2 = 3 EM CLASS 2 = 8 NOFA2 PROC3 = 4 EM CLASS 3 = 9 EOC PROC2 = 5	N/A	1
ANEFN	Emitter File No. ($0 \leq$ ANEFN ≤ 127)	N/A	1
ANPTR	Pointer to Candidate List	N/A	N/A
ANSTY	Scan Type of Emitter (see CDBDD for codes)	N/A	N/A
ANSPR	Scan Period of Emitter	msec	1/4

Figure 1b. (Scan) Analysis Return Message Format

Word	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	ANMNO															
1	ANNW															
2	ANRMC								ANEFN							
3	ANPTR															
4	AN AW											AN DI	AN CA	AN PA	AN FA	AN SA
5	AND1															
6	AND2															
7																
8																
9																
10																
11																
12																
13																
14	AND1Ø															
15	AND11															

Figure 2a. Analysis Request Message Format

Field	Description	Units	LSB
ANMNO	Executive Message No. (= 1)	N/A	1
ANNW	No. of Words in Message (= 3)	N/A	1
ANRMC	Return Module Code NEPROC2 = X'Ø1' NEPROC3 = X'Ø2' NOFA2 PROC2 = X'Ø3' NOFA2 PROC3 = X'Ø4' EOC PROC2 = X'Ø5' EOC PROC3 = X'Ø6' EOC PROC4 = X'Ø7' EM CLASS 2= X'Ø8' EM CLASS 3= X'Ø9'	N/A	N/A
ANEFN	Emitter File No. (0 ≤ ANEFN ≤ 127)	N/A	1
ANPTR	Pointer to Candidate List	N/A	N/A
ANAW	Analysis Wanted Code Ø = NO ANAL blank 1 = ANAL	N/A	N/A
ANDI	Deinterleaving Analysis Request Ø = None 1 = DO DI ANAL	N/A	N/A
ANCA	Contemporaneous Analysis Request Ø = None 1 = DO CA ANAL	N/A	N/A
ANPA	PRI Analysis Request Ø = None 1 = DO PRI ANAL	N/A	N/A
ANFA	Frequency Analysis Request Ø = None 1 = DO FREQ ANAL	N/A	N/A
ANSA	Scan Analysis Request Ø = None 1 = DO SCAN ANAL	N/A	N/A
AND1	Not Used in Priority 1 Software		
.	.		
.	.		
.	.		
.	.		
AND 'N'	Not Used in Priority 1 Software		

Figure 2b. Analysis Request Message Format

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	MNO															
1	NW															
2	NOT USED								EFN							
3	NOT USED															
4																
5																
6																
7																
8																
9																
10																
11																
12																
13	NOT USED															

Figure 3a. Classification Message Format

Field	Description	Units	LSB
MNO	Executive Message No. (= 9)	N/A	1
NW	No. of Words in Message (= 1)	N/A	1
EFN	Emitter File No. ($0 \leq EFN \leq 127$)	N/A	1

Figure 3b. Classification Message Format

Word	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ANMNO															
1	ANNW															
2	D	NOT USED							ANEFN							
3	NOT USED															
4																
5																
6																
7																
8																
9																
10																
11																
12																
13	NOT USED															

Figure 4a. Update Message Format

Field	Description	Units	LSB
ANMNO	Executive Message No. (= 7)	N/A	1
ANNW	No. of Words in Message (= 1)	N/A	1
ANEFN	Emitter File No. ($0 \leq$ AEFN ≤ 127)	N/A	1
D	Deletion Flag 1 = Emitter ANEFN has been made inactive Ø = Normal Update Message	N/A	N/A

Figure 4b. Update Message Format